BILINEAR IDENTIFICATION OF A BINARY DISTILLATION COLUMN

A thesis presented for the degree of
Doctor of Philosophy in Chemical and Process Engineering
in the
University of Canterbury,
Christchurch, New Zealand.

by
P.W.M. Janssen
1986

ABSTRACT

Low order bilinear models of chemical processes, suitable for control applications over a wide operating region, were identified.

A case study on a simulated heated tank system showed that there was considerable potential for the use of bilinear models for chemical processes in which mass or energy balance equations have a bilinear structure. Bilinear models were more accurate than linear models at fitting the simulated system's behaviour over a wide range of operating points. The U-D identification algorithm used, proved to be both robust and reliable.

Multi-input multi-output bilinear models of both a simulated and an experimental binary distillation column were identified using the U-D algorithm. Two factors affected the trend in the column parameters; the turn-down of the column and the shifting of the composition profile along the column. The bilinear models were good at following changes due to the turn-down, but not the large fluctuations due to changes in the composition profile. This limitation was due to the simplified liquid/vapour equilibrium relation inherent in the bilinear model, which was invalid if there were large fluctuations in composition on any of the plates.

ACKNOWLEDGEMENTS

I would like to thank Dr. R.M. Allen for his help and advice during this work.

Many thanks to the technicians of the Chemical and Process Engineering department for their efforts in constructing equipment.

This work was financially supported by a research scholarship from the New Zealand Energy Research and Development Committee.

Lastly, I would like to thank my fellow postgraduate students for their support during the long haul.

CONTENTS

Page

Page

Page

Page

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Over recent years much work has been done on the identification and control of distillation columns. This attention has been warranted by the fact that distillation processes are a major energy consumer in the chemical process industries. Reducing this energy usage can be tackled on a number of different levels. At the highest level one can attempt to optimise the set points for the distillation column as part of an overall plant strategy, this usually being considered as a steady state problem. The next level down, dealt with in this work, is to control the process as closely as possible to the set points. This closer control allows a decrease in the margins between set points and product specifications, thereby reducing energy consumption.

It is desirable to maintain this close control even when faced with large changes in operating conditions due to load variables or changes in set points. The techniques used must also be robust and capable of dealing with such real life problems as measurement and process noise, dead time and modeling errors.

1.2 THE DISTILLATION PROCESS

A distillation column is a non-linear multi-input multi-output process. The process inputs are feed composition, feed thermal condition, and feed, distillate, reflux, bottoms and steam flowrates. The process outputs are reboiler and reflux accumulator levels, and distillate and bottoms compositons.

Shinsky (1967) showed that to control the inventory of the column it is necessary to control at least one of the levels with either the distillate or bottoms flowrate. The relative gain matrix can be used to show which variables to use to control the column levels, and the interaction measure developed by Bristol (1966) can give a quantitative idea on the difficulty of control.

In this work, the effect of the two level control loops was regarded as an integral part of the column process. This resulted in the number of inputs and independent outputs each being reduced by two. The two remaining outputs were the distillate and bottoms composition. The inputs were divided into control inputs and disturbance inputs. The operator has no control over the latter in a realistic situation, but at best, they can be measured. The control inputs were the reflux and steam flowrates, and the disturbance inputs were feed flowrate, composition and thermal condition.

1.3 IDENTIFICATION AND CONTROL

It is essential for all control applications to have a model of the process to be controlled, whether it be a Bode diagram, Laplace transfer function or a state space type model. Even for ON/OFF control it is necessary to know the direction of the process response to a switching of the input.

The choice of model is often made according to the control procedure being used. Most researchers in the past have limited themselves to fitting time invariant linear models to distillation columns (Webb and Edgar (1979), Defaye et al (1977)). This was done for ease of identification and for ease of design of the control system rather than because distillation columns are inherently linear processes. In general, their results have only been satisfactory for small upsets and for when the column was operated under conditions close to those under which it was

identified. The reason for these limitations was the narrow validity region of the linearisation of the column response.

There are two different approaches which attempt to overcome this problem. In the black box approach of linear adaptive control, no *a priori* knowledge is used. In the non-linear modelling approach, as much *a priori* knowledge is used as can be assimilated into the strategy. Both approaches have intuitive appeal. The black box approach is portable and requires a minimum effort. The non-linear modelling approach should result in better control because more information is being used. This view is supported by the works of Aris et al (1981) and Goodwin et al (1981) which both show an improvement in control if an appropriate non-linear model is used in deriving the control action rather than a linear model.

An alternative to the two extremes mentioned above is to use *a priori* information to deal with the strongest non-linearities and to use on-line parameter estimation, possibly intermittent, to deal with smaller effects and with time varying parameters.

1.4 ADAPTIVE CONTROL

For adaptive control, the parameters of a model, usually linear, are continuously updated and then used to adjust the controller tunings (Astrom and Wittenmark (1973), Clarke and Gawthrop (1975)). Self-tuning regulators (STR's) handle non-linearities by treating them as time-varying parameters, but they have a number of major difficulties. To continuously update the plant parameters in the identified model, STR's employ a forgetting factor in their parameter estimation algorithm which has the effect of weighting the most recent measurements more heavily. In practice, it means that to prevent STR's learning the wrong model either the plant must be continuously excited, variable forgetting factors employed, or the parameter estimation

turned off when not required. Secondly, the speed with which a STR can follow a change in parameters decreases as the number of parameters increases. Hence, if there is a large change in column operating point, the STR needs to re-evaluate the large number of parameters involved in a multi-input multi-output system. During this time the column's transient response can be less than desirable.

Recently, both Dahlqvist (1981) and Morris et al (1981) have applied multivariable self-tuning regulators to experimental distillation columns. Dahlqvist used least squares parameter estimation and a minimum variance controller. Morris et al used a UDU algorithm for parameter estimation and a minimum variance controller.

1.5 NON-LINEAR MODELS

There is obviously some advantage to treating the column as a time-invariant non-linear system. The difficulty lies in finding a form of non-linear equation which satisfactorily describes a distillation column, but is simple enough to be used for control without too high a computational penalty.

It is possible to model the behaviour of distillation columns rigorously with mass and energy balances. As shown in Holland and Liapis (1983), this results typically with between one and three ordinary differential equations for each plate depending on the assumptions made. These equations can be integrated by one of the available packages. It is conceivable that one could combine this rigorous model with a direct search technique to provide a form of optimum control. But the integration and search techniques are so time consuming that with the present level of computer technology it could not be used in real-time. It could however, be used to generate a schedule of controller settings off-line. In this work we use such a rigourous model to generate

input/output data for the evaluation of parameters in more simplified models.

Espana (1977) showed that by making several simplifying assumptions the rigorous model could be reduced to a continuous state space bilinear model which uses the compositions on all trays as its state vector. This type of model has received considerable interest by researchers in other fields and there is considerable theoretical support available. Espana then reduced the order of the model by considering the column as three interacting tanks which resulted in a 3-term state vector and 8 parameters requiring indentification.

1.6 THIS WORK

The objective of this work was to find a non-linear model capable of describing the dynamics of a distillation column well enough for control applications over a wide operating region, but sufficiently simple to allow for straightforward calculation of control action. Implicit in this objective was the need to develop a robust and computationally inexpensive identification technique.

Chapter 2 describes the experimental binary distillation column used for this work. This column was extensively modified by the author to improve the control of column inputs and the measurement of column variables. The most significant of these changes were the developments of capacitance cells and a computer interface. The cells gave instantaneous measurements of feed and product compositions. The interface provided the capability to perform on-line data acquisition and analysis.

The dynamics of this column were simulated in the next chapter in order to provide a tool with which to develop identification techniques. The results of this simulation

compared favourably with those of other researchers. Although distillation column simulation is a well researched field, this simulation had a number of unusual features. The number of differential equations was reduced to two for each plate, without making any restrictive assumptions. The initial steady state solution was found by performing a dynamic simulation until it converged. This steady state simulation had been modified to reduce stiffness and to increase the rate of convergence.

Chapters 4 and 5 provide summaries of the fields of identification and bilinear systems, respectively. The methods described here were used in the later chapters. The emphasis was placed on recursive identification techniques, which could eventually be used in real-time to track time-varying parameters. Bilinear systems were chosen because distillation columns are compartmental systems, which fall naturally into this class of non-linear systems. The necessary conditions for an input signal to persistently excite a bilinear system were derived in Chapter 5.

Research on the identification of simulated bilinear systems was limited and that for experimental systems was rare. Chapter 6 deals with the identification of a simulated bilinear heated-tank system, which was performed to gain expertise in this area. The U-D algorithm was used to identify bilinear discrete difference models of the system, and its robustness in the presence of modelling error and measurement noise was studied.

In Chapter 7, distillation columns were shown to fall in the class of systems termed bilinear. The U-D algorithm was used to identify multi-input multi-output linear and bilinear models of the simulated distillation column described in Chapter 3. An important aim of this chapter was to test if the simplified diagonal bilinear models could adequately represent distillation column behaviour. The identified models were then used to

predict the column response to a different input function. The use of the simulated column avoided the difficulties associated with experimental data and facilitated the study of the validity regions of the different types of model.

In the next chapter, the same techniques were applied to the experimental column described in Chapter 2. This confirmed the results of the previous chapter and validated the approach of using a dynamic simulation as a development tool. However, it did show up some deficienies in the experimental column and the need to take more environmental effects into account. The robustness of the identification technique was demonstrated by its ability to deal with experimental data.

CHAPTER 2

EXPERIMENTAL COLUMN

2.1 COLUMN SPECIFICATIONS

The distillation column used in this work had previously been upgraded and used by Wilson (1979). This previous work had included a redesign of the column internals, a resizing of the reboiler and condenser, and the design and manufacture of vane pumps with variable speed drives. The column instrumentation had been improved, with temperature measurements on each plate, differential pressure measurements of the internal liquid levels, and the commissioning of a refractometer to measure composition. The column had then been interfaced to a microprocessor used as a stand alone controller.

A summary of the column specifications is given here, a more detailed description can be found in the work of Wilson.

Operation	-	atmospheric
Feed	-	mixture of methanol and water enters on 5th plate from top
Column	-	QVF glass, 300mm x 225mm diameter sections
Plates	-	8 sieve plates spaced at 400mm 223 x 3mm diameter holes per plate 19mm high entry and exit weirs
Reboiler	-	thermosyphon shell and tube type steam on shell side 100kW heat duty
Condenser	-	3 off QVF glass coils 98kW total heat duty

For this work, the column, plates, reboiler and condenser were left unchanged from that used by Wilson. The pumps and storage tanks were modified to give the operator some control over feed composition and flowrate. The instrumentation was upgraded to provide more reliable measurements of composition and flowrate, and the microprocessor and its software were changed to give a more user-friendly environment. The new column layout is shown in Figure 2.1 .

2.2 MECHANICAL MODIFICATIONS

A new feed pump was installed to bring the feed flowrate under remote control. This was a vane pump of the same design (Wilson 1979) as that used for the other main column flows and was driven by a three phase motor with a variable frequency controller. The teflon vanes and cams used in these pumps were dimensionally unstable at moderate temperatures, causing the pumps to seize. All the vane pumps were upgraded to use graphite vanes running on a brass cam.

In the original layout, a single tank was used to hold the feed, and the distillate and bottoms products were returned to the same tank to be remixed. This single tank has been replaced by four separate, interconnected tanks. Two tanks were mounted at a higher level and these was used to collect the distillate and bottoms products respectively. A manifold of four solenoid valves was used to fill the two lower feed tanks with a combination of the two products. Lastly, two solenoid valves on the exits of the lower tanks gave a choice from which to draw the feed, thereby giving the operator some control on feed composition. Stirrers were also installed on the top of the two feed tanks.

A Saunders diaphragm valve was initially used to control the steam flowrate to the reboiler chest. This exhibited a large degree of hysteresis during calibration, and replacement of the

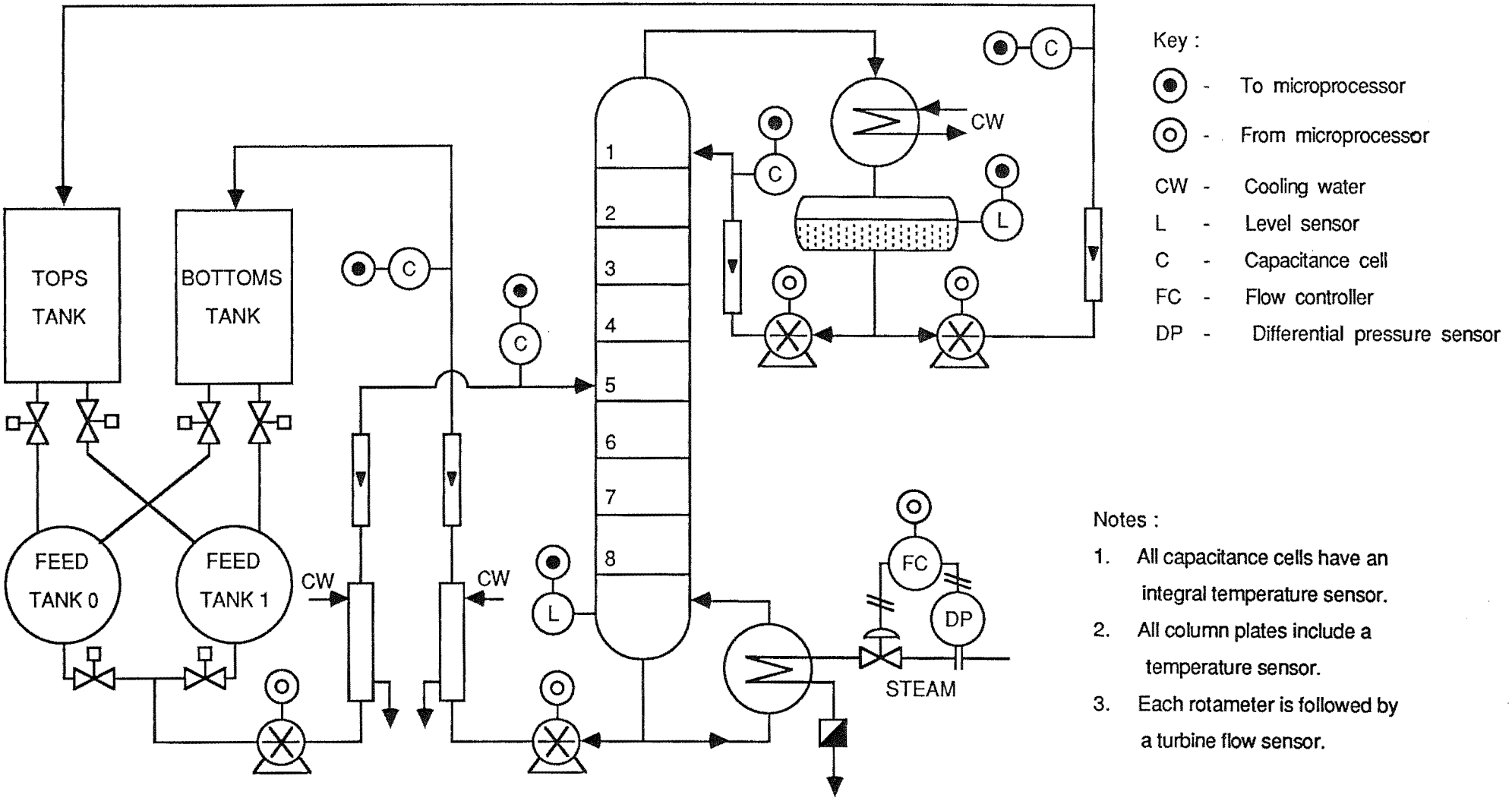


Figure 2.1 Column Layout

diaphragm and the tightening of all clearances made only a partial improvement. The valve was replaced by a Honeywell V5011A equal percentage control valve with a Cv of 4.

The initial layout had the positions of the orifice plate and control valve inverted from that shown in Figure 2.2. In this layout, the downstream pressure of the orifice plate was the same as the pressure in the reboiler chest, which varied with the boiling point of the liquid in the reboiler. As a result, the pressure drop across the orifice plate varied with both steam flowrate and bottoms composition. This meant that the pressure drop across the orifice plate had to be compensated for the reboiler pressure, in order to obtain the steam flowrate.

The steam flow control loop was rearranged as shown in Figure 2.2. In this layout, the upstream pressure of the orifice plate was the same as the outlet pressure from the pressure reduction valve. The controller was retuned manually. A proportional band of 500 and an integral time of 0.1 minutes were found to give acceptable results. The loop was then calibrated as shown in Appendix C.1.

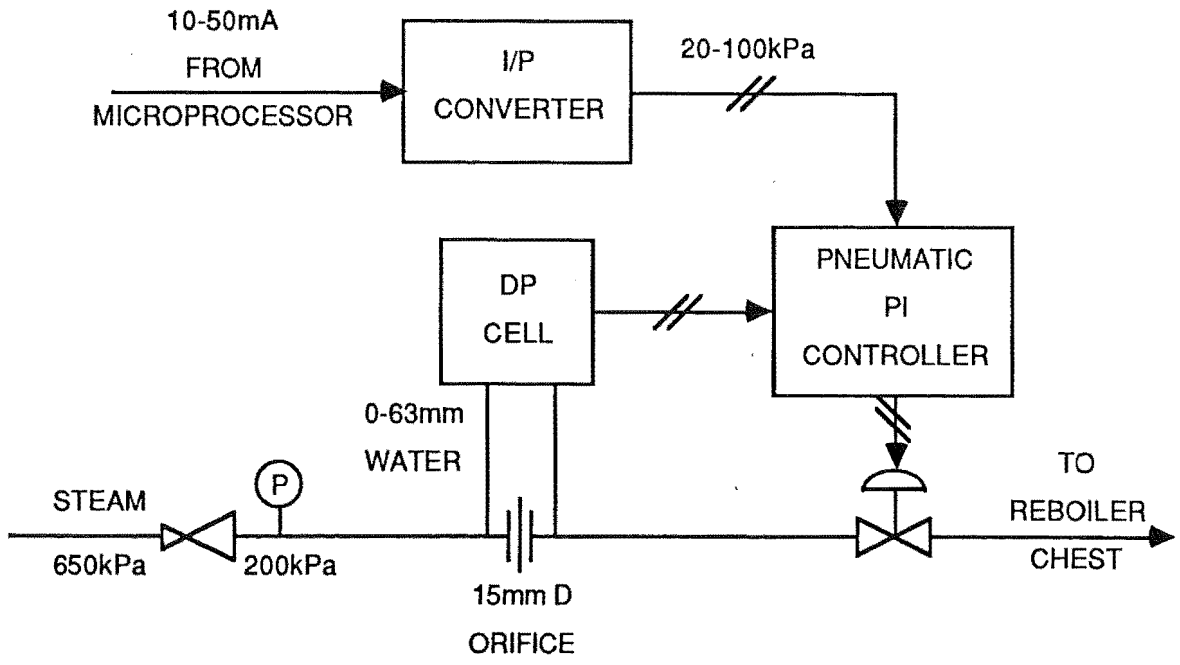


Figure 2.2

2.3 INSTRUMENTATION

2.3.1 TEMPERATURE MEASUREMENT

The liquid temperatures close to the exit weirs on all the column plates, and the temperatures at a number of other points around the column were measured using National Semiconductor LX5700 integrated circuit temperature transducers mounted at the end of stainless steel probes. These transducers were obsolete and when they became unreliable were all replaced with the newer LM335 transducers mounted on the same probes. New amplifier circuits were manufactured for each of the twelve transducers, and scaled to provide the standard 0 to 10V for 60 to 100°C on the transducers mounted on the plates, and 0 to 100°C for all other transducers.

2.3.2 PRESSURE MEASUREMENT

Pressure transducers were used in conjunction with pneumatic bubblers to measure the liquid levels in the reboiler and condenser, and the levels in the four storage tanks. The original National Semiconductor LX1601DF pressure transducers had unacceptable long term drift for these applications, and were replaced by six Honeywell 140PC05D transducers. The transducer outputs were filtered by passive first order filters, with time constants of 6 seconds, to remove the noise generated by the bubblers, and were then amplified and scaled to provide 0.612 to 10V for 0 to 1 metre of water. Reliable measurements of low pressure differentials were required for the control of the internal column levels, and the 0.612V output at zero differential pressure meant that if there was any zero drift the reading would still be on scale.

2.3.3 FLOWRATE MEASUREMENT

Previously, the liquid flowrates had been inferred by the speed of the vane pumps, which in practice were found to be far from ideal positive displacement pumps. The vane pumps would not pump at low speeds because there was considerable leakage past the vanes. The flowrates for the same speed varied with time and liquid temperature, as shown by the in-line rotameters.

Moray turbine meters were installed in the feed, reflux, distillate and bottoms lines. These meters had been modified by mounting the turbine inside glass tubes rather than the original acetal tubes which were unable to resist methanol. The output signal from these meters was a square wave with the frequency almost proportional to the volumetric flowrate. The outputs were connected to frequency counters on the column microprocessor and each meter was individually calibrated as shown in Appendix C.2.

2.3.4 COMPOSITION MEASUREMENT

Composition had previously been measured by an on-line refractometer. This instrument measured the fraction of light that was reflected from the glass/liquid interface which is a function of the refractive index, rather than measuring the angle of refraction directly. This measurement drifted slowly if the glass prism became fouled. The relationship between refractive index and composition for the water/methanol system has a turning point close to the feed composition, which made it difficult to get reliable measurements of the feed composition. The relationship is strongly dependent on temperature and both the sample liquid and the refractometer cell required elaborate temperature control. The refractometer sampled each of the column flows through a manifold of solenoid valves, giving intermittent readings and a one minute delay for each measurement in order to reach steady state.

Other researchers (Svrcek 1967, Sastry et al 1977) have successfully used capacitance cells to measure composition. The function relating dielectric constant to liquid composition at different temperatures is given in Appendix E.1. The temperature dependence is weak and the range of dielectric constant is large.

A flat plate capacitance cell was developed with two 75mm x 50mm stainless steel plates, initially spaced at 2mm. These dimensions were similar to those used by Akerlof (1937) to generate the data often quoted in the literature, and the overall cell capacitance was such that small stray capacitance effects could be ignored and circuit board layout was not critical. An LM335 temperature sensor was included on the cell to make temperature corrections to the composition measurements. Three of these cells were installed giving essentially instantaneous measurements of feed, distillate and bottoms composition.

Capacitance is usually measured with a capacitance bridge but this method is not suitable for automatic operation. The cell was used as the capacitor in an LC oscillator which initially had a frequency of 4 Mhz. The circuit diagram for the LC oscillator is shown in Figure 2.3. The cell was in series with a blocking capacitor and in parallel with a resisitor, which meant that the AC voltage across the cell had an average of zero volts which prevented polarisation, and a peak of less than the decomposition voltage of water which prevented electrolysis. The sine wave output from the oscillator was put through a Schmidt trigger and divided by a factor of 256 before being put into a counter on the column microprocessor.

During commissioning, these sensors were erratic and experienced unacceptable zero drift, especially when measuring feed and bottoms composition. This was due to the presence of ions in the solution and a small amount of oil contaminating the solution. The ions caused a decrease in the solution resistance affecting

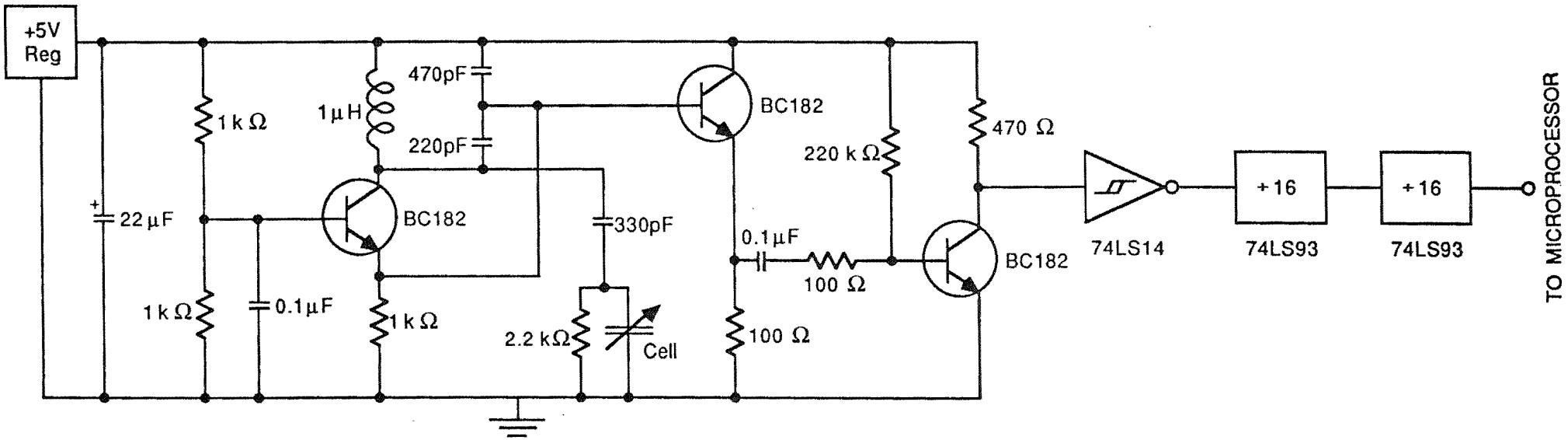


Figure 2.3 Capacitance Cell Oscillator

the oscillator. Oil, on the other hand, deposited in a fine film over the cell plates, changing the effective plate spacing. The contents of the column were drained, and then distilled in a 20 litre batch still to remove as many contaminants as possible. The plate spacing was increased to 4mm, reducing the effect of plate fouling, and the oscillator frequency was increased to 12MHz, since at higher frequencies the capacitive reactances overshadows the resistance.

Each of the capacitance cells was calibrated using solutions of known dielectric constants and polynomials, as shown in Appendix C.3, were fitted relating dielectric constant to oscillator frequency.

2.3.5 DATA AQUISITION

A Burr Brown SDM851 hybrid data acquisition module was used to measure the outputs of the temperature and pressure transducers. This module contained a fast 12-bit successive approximation type analogue to digital converter, preceeded by a 16-channel multiplexer. Each channel was filtered by an active second order filter to eliminate high frequency and mains-induced noise.

2.3.6 CONTROLLER OUTPUTS

The variable-speed pump controllers and the steam flowrate control loop received their set points from a 5-channel, 8-bit, digital to analogue converter based on Motorola MC3408 integrated circuits. At first, the outputs were scaled to provide the standard 0 to 10V range, but the limited resolution of the D/A converters meant that the flowrate control loops had appreciable limit cycling. The D/A outputs were then individually scaled so that their full range reflected the normal operating range of the variable-speed pumps and the steam flowrate.

A 6-channel, TTL to 240V interface was installed to switch the solenoid valves on the feed and product tanks. This interface used optotriacs to directly switch the 240V to the valves, and included a capacitor and resistor network to improve power factor and keep the triacs within their current specifications.

2.4 MICROPROCESSOR

All the column instrumentation was interfaced to a microprocessor, which in turn, was connected to the departmental VAX mini-computer. A schematic of the interfacing is given in Figure 2.4.

The microprocessor was based on an AMI EVK300 prototyping board which was connected to several other boards through a wire-wrapped back plane. The original Motorola M6800 microprocessor chip on the prototyping board was replaced by the later M6809 microprocessor. This new chip had a much improved instruction set, in particular it supported the indirect addressing mode, and had an extra index register and stack pointer, which were useful for writing efficient structured programs. This change required the manufacture of a small "piggy back" board which included logic for the different pin configuration and meanings on the new chip.

The AM9511 arithmetic processor board was retained. This stack orientated device provided fast hardware arithmetic and reduced the time taken by the microprocessor to do complex calculations. The timing potentiometers were adjusted to be compatible with the M6809 chip which had different requirements to the M6800 for its clock signals. The contents of the on-board EPROM were changed to reflect the shift away from the threaded code used by Wilson (1979) to simple subroutine calls to drive the arithmetic processor.

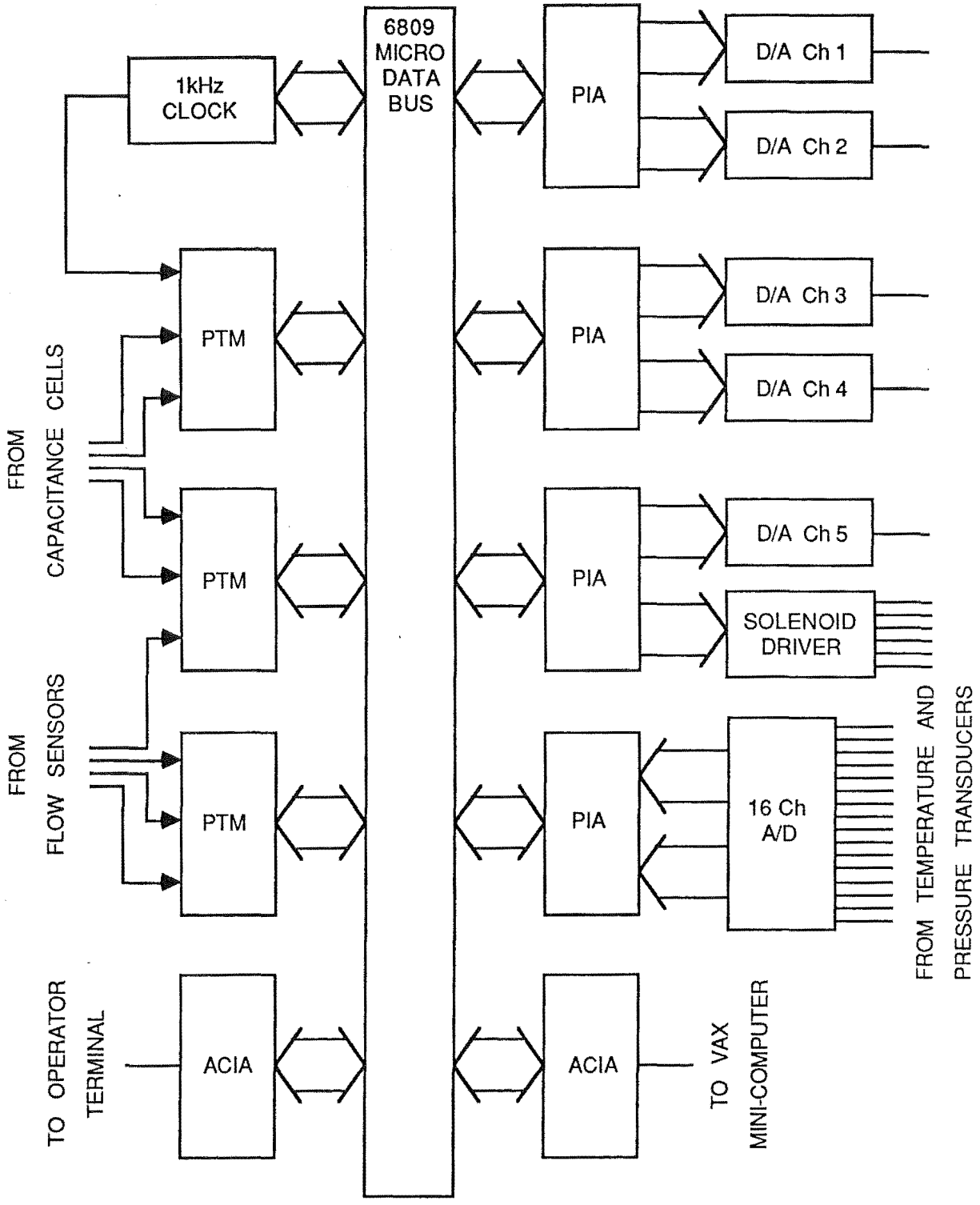


Figure 2.4 Microprocessor Interface

Three new circuit boards were manufactured and installed in the back plane. These contained additional memory and peripheral devices giving the microprocessor the following specifications :

9K read/write memory (static RAM)
8K erasable/programmable read only memory (EPROM)
5 16-bit wide parallel ports (PIA)
3 programmable timer/counter modules (PTM)
2 serial ports (ACIA)

2.5 OPERATING SOFTWARE

2.5.1 OVERVIEW

The column operating software consisted of three parts. The microprocessor monitor and the COLSYS column operating program were written in 6809 assembler language and were cross-assembled on the VAX. The third part, the operator interface and high level control program, was written in FORTRAN on the VAX.

This configuration was ideal because it meant all low level tasks were performed using efficient assembler language programs which did not need to be modified often. Meanwhile, higher level tasks, which inherently require more complex logic and calculations, were done in high level language which was more suited to the task and was much simpler to modify.

2.5.2 PR9MON MONITOR

The PR9MON monitor was based on the widely available PSYMON monitor, but with changes to support the local hardware configuration and to download programs from the VAX. It was programmed into an EPROM and control was passed to it whenever the microprocessor was powered up. It included a number of low level routines which could be called by other programs and these were used by COLSYS.

The instruction set is listed in Appendix A.1, as well as the procedure for downloading programs from the VAX.

2.5.3 COLUMN OPERATING PROGRAM

The COLSYS operating program was downloaded from the VAX using PR9MON. It handled all low level tasks such as frequency counting, simple PI control of levels and flows, and switching solenoid valves. If necessary, the column could be operated from the terminal using COLSYS alone, but the microprocessor was normally connected to the VAX. COLSYS made extensive use of the APU and a preassembler was written to facilitate the inclusion of APU floating point numbers directly into the assembler language program.

When requested, COLSYS would pass selected data up to the VAX where it was processed using programs written in high level languages. The program on the VAX could then request COLSYS to make a change in the set point of one of its controllers or change the state of a solenoid valve.

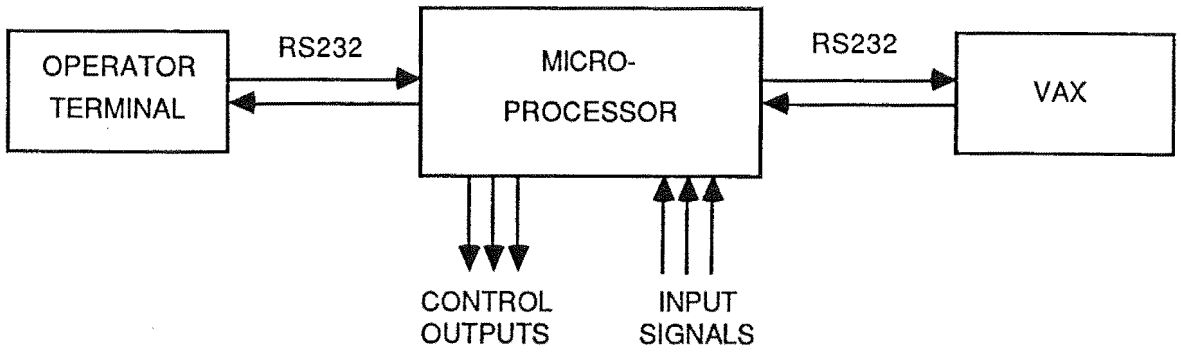


Figure 2.5

The microprocessor was connected as shown in Figure 2.5 with one serial line going to the terminal and another going to the VAX. Under normal circumstances the microprocessor was transparent both to the VAX and the operator. However any string of characters beginning with the hash character "#" and ending in

carriage return was intercepted by COLSYS and treated as an instruction to COLSYS. A description of all the instructions can be found in Appendix A.2.

The priority which COLSYS attached to its different tasks was determined by the following interrupt structure:

FIRQ	-	Serial line input
IRQ	-	Real-time clock and control functions
Non-interrupt	-	Serial line output and instruction handling

The highest priority was given to the input of characters on the two serial lines from where they were stored in first in/first out (FIFO) buffers. This was necessary because several characters could arrive during the time taken to do the control functions and these would otherwise be lost. The control functions were done under a lower priority interrupt which was generated using a programmable timer acting as a real-time clock. When the microprocessor was not in either interrupt mode, it checked and cleared the FIFO buffers, and carried out any instructions. The resultant effect was that it appeared to the operator as if he had all the microprocessor facilities at his disposal, while the task of controlling the column was done in the background. The IRQ interrupt service routine performed the following tasks :

1. Reset 1Hz real-time clock.
2. Read flow sensor and capacitance cell frequencies.
3. Average cell frequencies.
4. Read A/D converter.
5. Increment software timers.
6. Check for any alarm conditions.
7. Calculate controller outputs.
8. Filter measurements.
9. Report measurements if requested.
10. Manipulate solenoid valves if necessary.

The measurements reported by COLSYS to the operator or to the VAX were filtered before being sent. The flow sensor frequencies and the temperature measurements were filtered using digital first order filters as shown in Appendix A.4. The reported capacitance cell frequencies were the arithmetic average of the frequencies for the previous 8 seconds.

There were four PI control loops implemented in the column operating software using the algorithm described in Appendix A.3. The reboiler and condenser levels were controlled by changing the outputs to the bottoms and distillate pumps respectively. The feed and reflux flows, as measured by the turbine flowmeters, were controlled with the outputs to the feed and reflux pumps respectively. The control algorithms included slew rate limiting on the outputs to prevent the variable speed motors from dropping out of synchronisation.

The software used to implement both the digital filters and the controllers was table-driven and made extensive use of the AM9511 APU for most of its calculations. The table-driven feature facilitated the addition of more filters and control loops by placing an additional entry in the driving table. The use of the APU kept the overheads of the interrupt service routine low.

2.5.4 OPERATOR INTERFACE

The operator interface consisted of a main line FORTRAN program called COLUMN and a set of FORTRAN subroutines. A listing of these is given in Appendix B.1.

COLUMN interacted with the operator, allowing him to see the state of important column variables and to change the column inputs. It also gave the operator the option of reading the

column inputs from a data file and writing the column outputs to another file.

The set of subroutines provided an interface between COLUMN and the column operating program, COLSYS, as well as real-time functions on the VAX. The subroutines called Data_acquisition and Control_out provided data acquisition from the column and the sending of control signals to the column. They included the calibrations of flow, composition and temperature sensors, and that of the steam loop. The former converted measurements of dielectric constant and temperature into composition using the algorithm given in Appendix B.2.

COLUMN was useful as a basis from which to develop programs for other purposes, and the subroutines only needed to be changed if the sensors were recalibrated.

CHAPTER 3

DYNAMIC COLUMN SIMULATION

3.1 INTRODUCTION

Rigorous digital simulation is an important tool in the study of distillation column dynamics. It has been shown since the early 1960's that it may give results that closely follow those of the actual plant (Huckaba et al 1963), but without the difficulties associated with obtaining experimental results, such as measurement of composition and flows, process noise, and the time taken to complete an experiment. Dynamic simulation can not replace experimentation, but when used together they provide a good development facility.

The general mathematical model of a distillation column, as shown in Section 3.3, consists of a coupled set of simultaneous algebraic and ordinary differential equations. The differential equations are the result of performing component mass, total mass, and energy balances over each plate. The algebraic equations are the result of equilibrium, efficiency, and plate hydraulics relations.

The classical approach to solving this set of simultaneous equations is to manipulate the differential equations so that they are in the state space formulation, where the differentials of the state variables are given as explicit functions :

$$\frac{d}{dt}(\mathbf{y}) = \mathrm{fn}(\mathbf{y}, \mathbf{u}, t) \qquad (3.1)$$

Further, it is preferable that the algebraic equations are explicit in terms of the state variables. This approach has the major advantage that most integration algorithms and packages assume that the problem has been formulated in such a way.

Restrictive assumptions or approximations are often required to rearrange the equations in state space form or to decrease the dimensionality of the model. The energy balance equation can be reduced to an algebraic equation by assuming that energy changes are instantaneous or by the approximation of a derivative by a finite difference expression. The total mass balance equation can be reduced to an algebraic equation by assuming constant molar holdups on the plates.

The earliest attempts at dynamic simulation (Huckaba et al 1963) used only the differential equations due to component mass balances. They found that simple integration algorithms, such as modified Euler method, or Runge-Kutta, gave satisfactory results. The inclusion of other differential equations in the model resulted in excessively long integration times. These algorithms selected their time step in accordance with the quickest time constant in the system, but the large differences in the sizes of time constants resulted in time steps that were very short in comparison to the slower time constants and the length of the overall simulation. This problem was recognised as stiffness by Tyreus et al (1975), and can be measured by the stiffness ratio, which is the ratio of the largest to the smallest time constant. Special integration techniques, such as that of Gear (1971a), were developed for such problems. These always use integration algorithms of the implicit type, and they increase the integration interval once the effect of the smaller time constants is negligible.

The approach described above has difficulty if the algebraic equations are implicit because a root finding problem needs to be solved for every evaluation of the derivative. In recent times, a second approach has been developed by Gear (1971b) and it solves the differential and algebraic equations simultaneously. This has been used by Gallun (1982) to model a large multi-component distillation column. The same Newton-Raphson technique that is used to iterate the corrector equation to convergence is used to simultaneously solve the algebraic equations. For distillation columns, this technique can take advantage of sparse matrix techniques to reduce the computational load. A similar technique is the semi-implicit Runge-Kutta method developed by Michelson (1976). This has a disadvantage in that it requires the re-evaluation of the Jacobian matrix for every time step, while Gear's method only re-evaluates it when necessary.

In this work, the explicit state space approach has been followed since it allowed the use of integration packages such as EPISODE and LSODE without modification. It is shown in the following sections that the equations for a distillation column can be arranged in explicit form without the common assumption of constant molar holdups or the approximation of the enthalpy balance by an algebraic equation. The number of differential equations was reduced to two per plate by recognising that for atmospheric binary systems, where the liquid and vapour enthalpies are functions of liquid and vapour compositions, the time differentials of enthalpy and composition are not independent variables.

3.2 ASSUMPTIONS

The following assumptions were used in the dynamic simulation and resulted in the model presented in the next section :

1. Negligible vapour holdup.
2. Liquid well mixed on plates.
3. Tray hydraulics follow Francis weir formula.
4. Constant porosity of liquid on plates i.e. $\beta = 0.65$
5. Constant and equal heat losses from each plate.
6. Total condenser.
7. Feed in liquid state.
8. Reboiler is one ideal stage.
9. Reboiler has constant volume.

The total condenser and liquid feed assumptions accurately reflect the experimental column used in this work, but these assumptions can easily be relaxed for other situations. The assumption that the reboiler constituted one ideal stage was verified by comparing the steady state temperature profile along the experimental column with that of the simulation.

The low vapour density and its high velocity (~1m/s) supported the first assumption. The constant liquid porosity was supported by the relatively flat nature of its relationship to the column flows in the normal column operating region (Smith (1963)). This assumption was also necessary to rearrange the algebraic equations in explicit form. The assumption of a constant volume reboiler was justified because the liquid level in the experimental reboiler was controlled by maintaining a fixed pressure differential between two closely spaced tappings situated at the top of the reboiler.

3.3 GENERAL MATHEMATICAL MODEL

3.3.1 PLATE MODEL

The equations describing the general i^{th} plate shown in Figure 3.1 are given below. A description of all the variables is

given in the nomenclature section at the end of this chapter. The terms in F, x_f and h_f are only relevant for the feed tray.

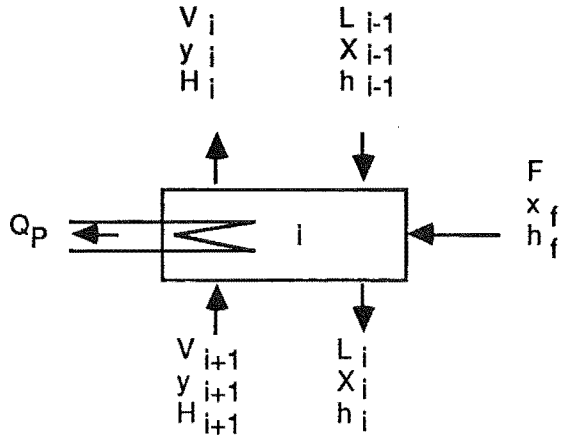


Figure 3.1

Overall mass balance :

$$\frac{dm_i}{dt} = V_{i+1} + L_{i-1} + F - V_i - L_i \qquad (3.2)$$

Component mass balance :

$$\frac{d(m_i x_i)}{dt} = V_{i+1}y_{i+1} + L_{i-1}x_{i-1} + Fx_f - V_i y_i - L_i x_i \qquad (3.3)$$

Enthalpy balance :

$$\frac{d(m_i h_i)}{dt} = V_{i+1}H_{i+1} + L_{i-1}h_{i-1} + Fh_f - V_i H_i - L_i h_i - Q_p \qquad (3.4)$$

Plate hydraulics from the Francis' weir formula :

$$L_i = \rho_i * l_w \left(\frac{h_{ow}}{4.34*10^{-4}} \right)^{1.5} \qquad (3.5)$$

where $$h_L = \beta(h_{ow} + h_w) \qquad (3.6)$$

and $$m_i = \rho_i * 10^3 * (h_L A_a + v_c) \qquad (3.7)$$

Murphree vapour efficiency :

$$E_i = \frac{y_i - y_{i+1}}{y_i^* - y_{i+1}} \qquad (3.8)$$

Liquid/vapour equilibrium :

$$y_i^* = k_i x_i \quad \text{where} \quad k_i = fn(x_i) \qquad (3.9)$$

3.3.2 CONDENSER MODEL

For the total condenser as show in Figure 3.2 the equations are as follows :

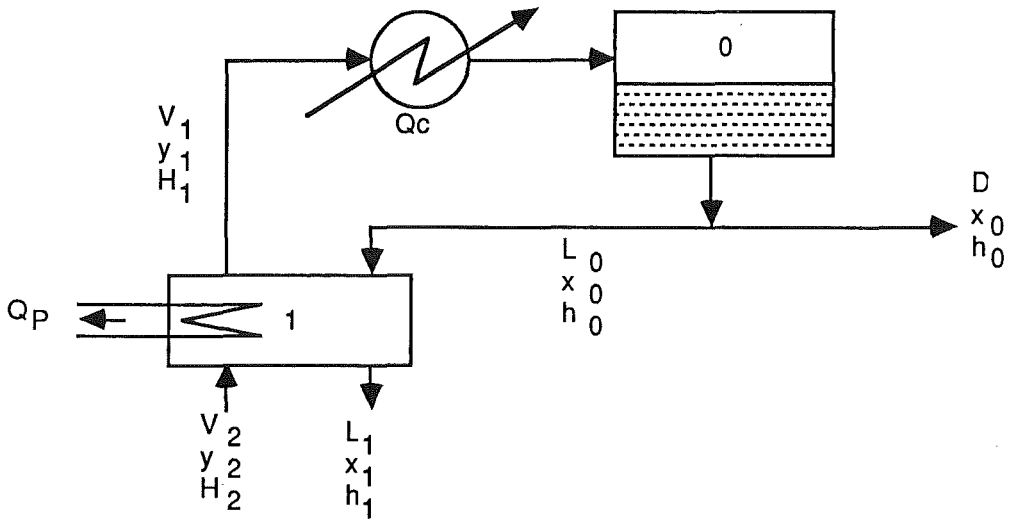


Figure 3.2

Overall mass balance :

$$0 = V_1 - L_0 - D \qquad (3.10)$$

Component mass balance :

$$\frac{d(m_0 x_0)}{dt} = V_1 y_1 - L_0 x_0 - D x_0 \qquad (3.11)$$

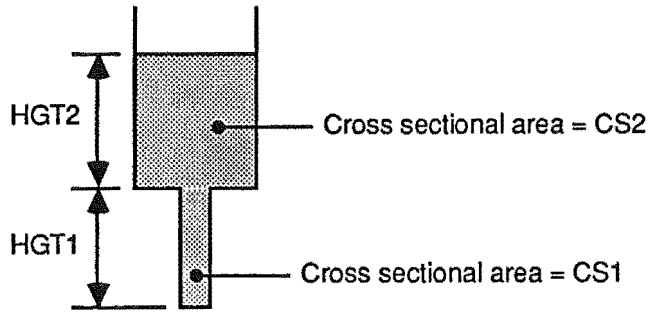


Figure 3.3

Molar holdup from constant head of liquid assumption :

$$m_0 = \rho * 10^3 * (HGT1*CS1 + HGT2*CS2) \qquad (3.12)$$

where

$$P_c = \rho * g * (HGT1 + HGT2) \qquad (3.13)$$

3.3.3 REBOILER MODEL

For the reboiler as shown in Figure 3.4 the equations are as follows:

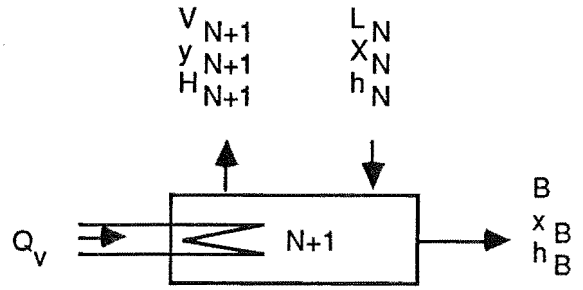


Figure 3.4

Overall mass balance :

$$\frac{dm_{N+1}}{dt} = L_N - V_{N+1} - B \qquad (3.14)$$

Component mass balance :

$$\frac{d(m_{N+1}x_{N+1})}{dt} = L_N x_N - V_{N+1}y_{N+1} - Bx_{N+1} \qquad (3.15)$$

Enthalpy balance :

$$\frac{d(m_{N+1}h_{N+1})}{dt} = L_N h_N - V_{N+1}H_{N+1} - Bh_{N+1} - Q_v \qquad (3.16)$$

Molar holdup from constant volume assumption :

$$m_{N+1} = \rho * 10^3 * v_r \qquad (3.17)$$

Liquid/vapour equilibrium :

$$y_{N+1} = k_{N+1}x_{N+1} \quad \text{where } k_{N+1} = fn(x_{N+1}) \qquad (3.18)$$

3.4 PROGRAM DETAILS

The dynamic column simulation program, listed in Appendix D, was batch orientated and used data files for all input and output functions. It required three input files :

1. Physical properties data.
2. Column parameters.
3. Column inputs.

It produced an output file containing the time response of the column to the disturbances defined in the input file. The initial steady state solution for the column was calculated using the first record of the input file. The subsequent records in the input file were used as the column inputs at discrete sampling intervals, usually 4 min, for the dynamic solution.

The program used the LSODE integration package developed by Hindmarsh (1980) to solve the systems of simultaneous differential

equations that are used in both the steady state and dynamic phases of the program.

LSODE assumed that the differential equations had been manipulated into a state space formulation. The user of this package must initialise the state vector before calling the LSODE routine and must supply a subroutine which evaluates the derivative of the state vector as a function of the state vector.

The LSODE package was used in this work with the integration method flag set to 22, which meant that the package estimated its own Jacobian using finite differences. This is discussed later under program performance.

3.5 STEADY STATE SOLUTION

The usual method of finding the steady state solution for a distillation column is to solve the steady state equations. These are a set of complex non-linear simultaneous algebraic equations and their solution necessitates an iterative root finding technique. In this work the steady state solution was found by running a simplified dynamic simulation until it converged to its ultimate values, but this simulation differed from that used in the dynamic solution. This approach had the advantage that there were no difficulties with convergence and that there was much commonality in the code used for the steady state and dynamic solutions.

For the steady state simulation, the tray, reboiler and condenser compositions were used as state variables. Plate hydraulics were ignored and algebraic enthalpy balances used. All holdups including trays, reboiler and condenser were set to unity to speed up convergence and reduce stiffness. These simplifications changed the trajectory but did not affect the ultimate values. The reduced stiffness meant that larger integration time

steps were used, and the reduced dimensionality decreased the calculations required for each time step.

The simplifications resulted in the equations given in Section 3.3 being reduced to the following expressions :

For the i^{th} plate :

Overall mass balance :

$$0 = V_{i+1} + L_{i-1} + F - V_i - L_i \tag{3.19}$$

Component mass balance :

$$\frac{dx_i}{dt} = V_{i+1}y_{i+1} + L_{i-1}x_{i-1} + Fx_f - V_iy_i - L_ix_i \tag{3.20}$$

Enthalpy balance :

$$0 = V_{i+1}H_{i+1} + L_{i-1}h_{i-1} + Fh_f - V_iH_i - L_ih_i - Q_p \tag{3.21}$$

For the total condenser :

Overall mass balance :

$$0 = V_1 - L_0 - D \tag{3.22}$$

Component mass balance :

$$\frac{dx_0}{dt} = V_1y_1 - L_0x_0 - Dx_0 \tag{3.23}$$

For the reboiler :

Overall mass balance :

$$0 = L_N - V_{N+1} - B \tag{3.24}$$

Component mass balance :

$$\frac{dx_{N+1}}{dt} = L_N x_N - V_{N+1} y_{N+1} - B x_{N+1} \qquad (3.25)$$

Enthalpy balance :

$$0 = L_N h_N - V_{N+1} H_{N+1} - B h_{N+1} - Q_v \qquad (3.26)$$

The total mass and energy balance equations were grouped in the following banded matrix form and solved by Gaussian elimination to give the column liquid and vapour flows.

$$\begin{bmatrix}
H_1 & h_1 & -H_2 & & & & & & & \\
1 & 1 & 1 & & & & & & & \\
 & -h_1 & H_2 & h_2 & -H_3 & & & & & \\
 & -1 & 1 & 1 & -1 & & & & & \\
\hline
 & & -h_{f-1} & H_f & h_f & -H_{f+1} & & & & \\
 & & -1 & 1 & 1 & -1 & & & & \\
\hline
 & & & & & & -h_N & H_{N+1} & h_{N+1} \\
 & & & & & & -1 & 1 & 1
\end{bmatrix}
\begin{bmatrix} V_1 \\ L_1 \\ V_2 \\ L_2 \\ \hline V_f \\ L_f \\ \hline V_{N+1} \\ L_{N+1} \end{bmatrix}
=
\begin{bmatrix} L_0 h_0 - Q_p \\ L_0 \\ -Q_p \\ 0 \\ \hline Fh_f - Q_p \\ F \\ \hline Q_v \\ 0 \end{bmatrix} \qquad (3.27)$$

The state vector was initialised assuming a linear temperature profile down the column. The integration interval passed to LSODE was progressively doubled until the state vector converged, this being checked with a sum of the squares of the change in temperature down the column since the previous integration interval. The subroutine to calculate the derivative

of the state vector for the steady state solution used the following procedure :

1. Use x_i's to calculate h_i's, k_i's and E_i's
2. Calculate y_i's
3. Use y_i's to calculate H_i's
4. Set up banded matrix and solve to find L_i's & V_i's
5. Calculate $\frac{dx_i}{dt}$'s

3.6 DYNAMIC SOLUTION

The dynamic solution used the plate, reboiler and condenser compositions, and the plate molar holdups as the state variables. The equations given in Section 3.3 were manipulated into an explicit form in order to use LSODE. Equations 3.2 to 3.4 give the three differential equations for a mass and energy balance over a general plate. These can be reduced to two differential equations per plate by using the algebraic equation given in Section 3.7 which relates saturated liquid enthalpy to composition, and by application of differential calculus.

The application of the chain rule resulted in :

$$\frac{d(m_i x_i)}{dt} = m_i \frac{dx_i}{dt} + x_i \frac{dm_i}{dt} \qquad (3.28)$$

$$\frac{d(m_i h_i)}{dt} = m_i \frac{dx_i}{dt}\frac{dh_i}{dx_i} + h_i \frac{dm_i}{dt} \qquad (3.29)$$

where $$\frac{dh_i}{dx_i} = fn(x_i) \qquad (3.30)$$

Hence Equations 3.2 to 3.4 can be rearranged to give :

$$\frac{dm_i}{dt} = V_{i+1} + L_{i-1} + F - V_i - L_i \qquad (3.31)$$

$$\frac{dx_i}{dt} = \frac{V_{i+1}}{m_i}(y_{i+1}-x_i) + \frac{L_{i-1}}{m_i}(x_{i-1}-x_i) + \frac{F}{m_i}(x_f-xi) - \frac{V_i}{m_i}(y_i-x_i) \qquad (3.32)$$

$$\begin{aligned} 0 = {} & V_{i+1}\left[(H_{i+1}-h_i) - \frac{dh_i}{dx_i}(y_{i+1}-x_i)\right] \\ & + L_{i-1}\left[(h_{i-1}-h_i) - \frac{dh_i}{dx_i}(x_{i-1}-x_i)\right] \\ & + F\left[(h_f-h_i) - \frac{dh_i}{dx_i}(x_f-x_i)\right] \\ & - V_i\left[(H_i-h_i) - \frac{dh_i}{dx_i}(y_i-x_i)\right] - Q_p \end{aligned} \qquad (3.33)$$

The differential equations, as shown above, have been manipulated to give the derivative of the state vector explicitly. The state vector was initialised using the results of the steady state solution. The system was then integrated for time intervals, usually 4 minutes, which were chosen to be similar to the sampling frequency used on the experimental column.

The column inputs were changed at the start of each time interval, using values from the input data file. The integration algorithm in LSODE was forced to restart with a first order

integration method at the beginning of each time interval if the new column inputs were different from those in the previous interval. This was necessary because the higher order integration methods used estimated values of high order derivatives which were invalid if there was a discontinuity in the input function.

The subroutine to calculate the derivative of the state vector for the dynamic solution used the following procedure :

1. Use x_i's to calculate h_i's, k_i's, E_i's and $\frac{dh_i}{dx_i}$'s
2. Calculate y_i's and hence H_i's
3. Evaluate L_i's
4. Evaluate V_i's
5. Calculate the derivative of the state vector

3.7 PHYSICAL PROPERTY DATA

The numerical data for the methanol/water system was taken from the work of Wilson (1979).

1. Equilibrium constants - A cubic spline was fitted through 41 equally spaced points. The data is given in Appendix E.2.

2. Saturated liquid mass densities

$$\rho = -0.2223 * x + 0.97 \quad [kg.l^{-1}]$$

3. Saturated liquid molar densities

$$\varrho = \rho * 1000.0 / (14.024 * x + 18.016) \quad [mol.l^{-1}]$$

4. Murphree vapour efficiencies

$$E = 0.436 * x^3 - 1.03 * x^2 + 0.786 * x + 0.614$$

5. Saturated liquid enthalpies

$$h = 16.47 * x^4 - 38.97 * x^3 + 33.93 * x^2 - 13.63 * x + 7.524 \quad [kJ.mol^{-1}]$$

6. Saturated vapour enthalpies

$$H = 1.49 * y^2 - 9.084 * y + 48.21 \quad [kJ.mol^{-1}]$$

7. Boiling points

$$temp = -0.240 * x^5 + 739.8 * x^4 - 874.2 * x^3 + 501.9 * x^2 - 162.7 * x + 99.93 \quad [^\circ C]$$

8. Liquid heat capacity

$$C_p = (0.67*10^{-7} * T + 0.0126)* x + 0.134*10^{-4} * T + 0.07467 \quad [kJ.mol^{-1}.^\circ C^{-1}]$$

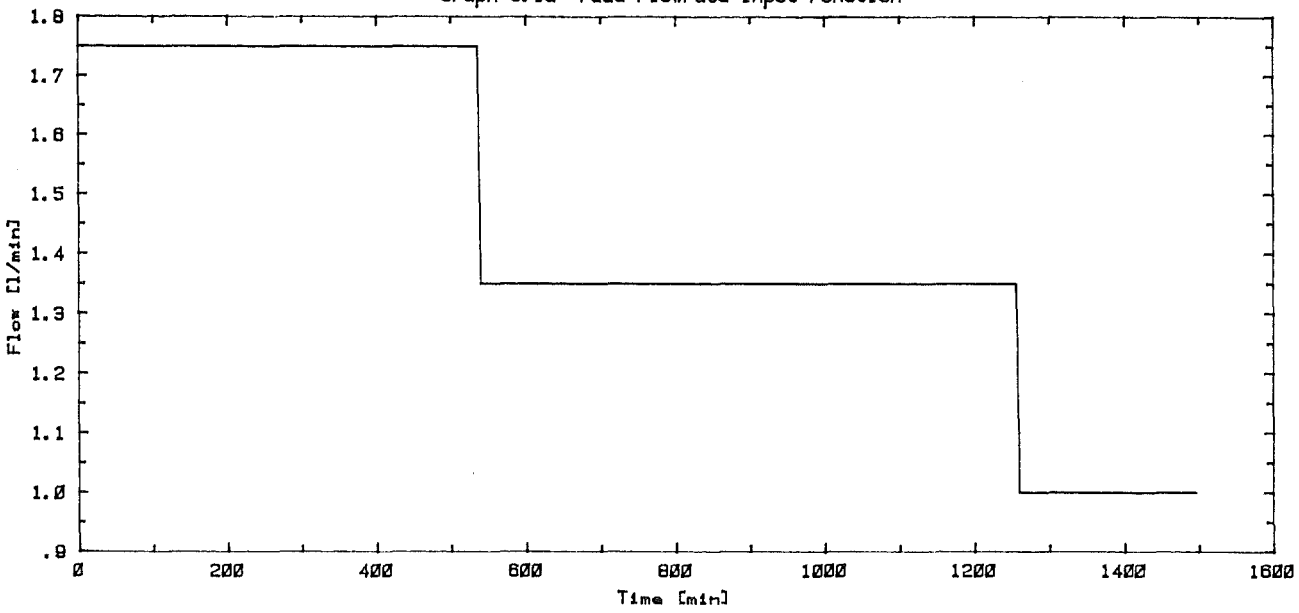
3.8 PROGRAM VERIFICATION

There were a number of parameters used in the simulation which were not known with certainty for the experimental column and were chosen so as to obtain a reasonable correspondence between simulated and experimental results. These were the molar holdup of the reboiler, and the heat losses from the column. It was expected that changing the holdups used in the simulation would mainly affect the time constants of simulation and the heat losses would effect the ratio of the distillate and bottoms flows and hence their composition.

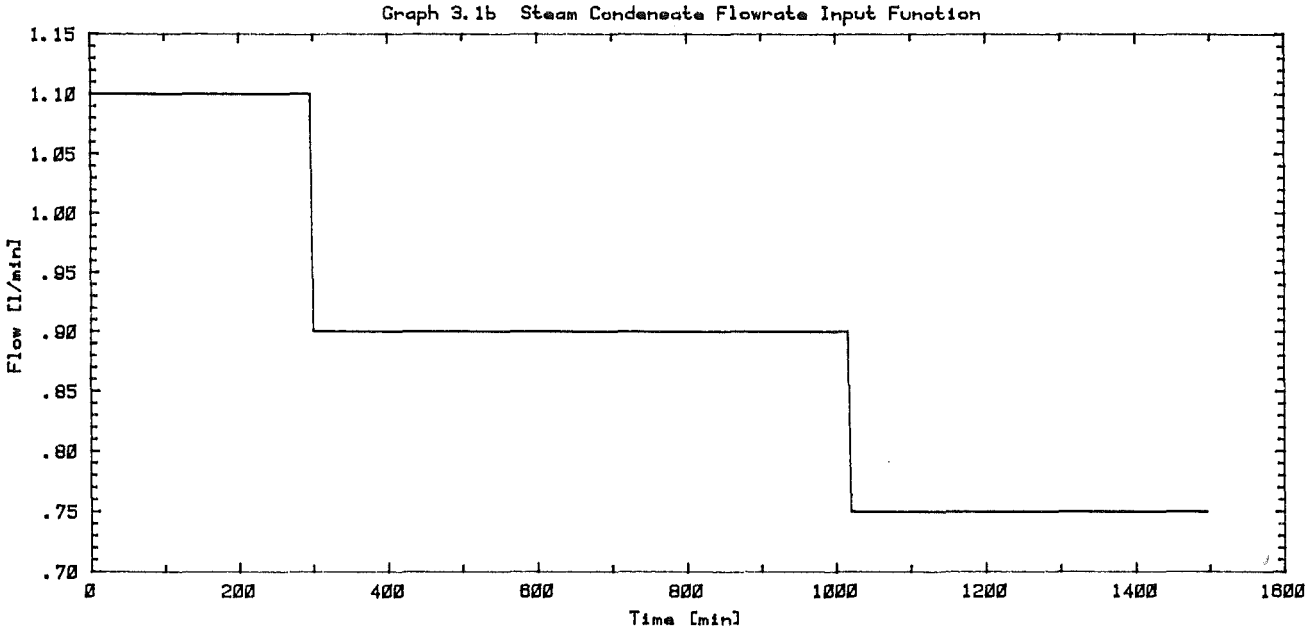
Two experimental runs were performed to aid in the verification of the simulation. Both runs used the same input functions shown in Graphs 3.1a to 3.1c in order to get an estimate of the repeatability of the results. This input function was chosen because it drove the column over almost its entire operating region. The resultant outputs for the two runs are shown in Graphs 3.2a to 3.3b. The simulation parameters were then tuned to obtain a similar input/output relation as shown in Graphs 3.4a and 3.4b.

The correspondence between the simulated and experimental responses was good and compared favourably with those of recent papers (Morris and Svrcek (1980), Kumar et al (1984)). The

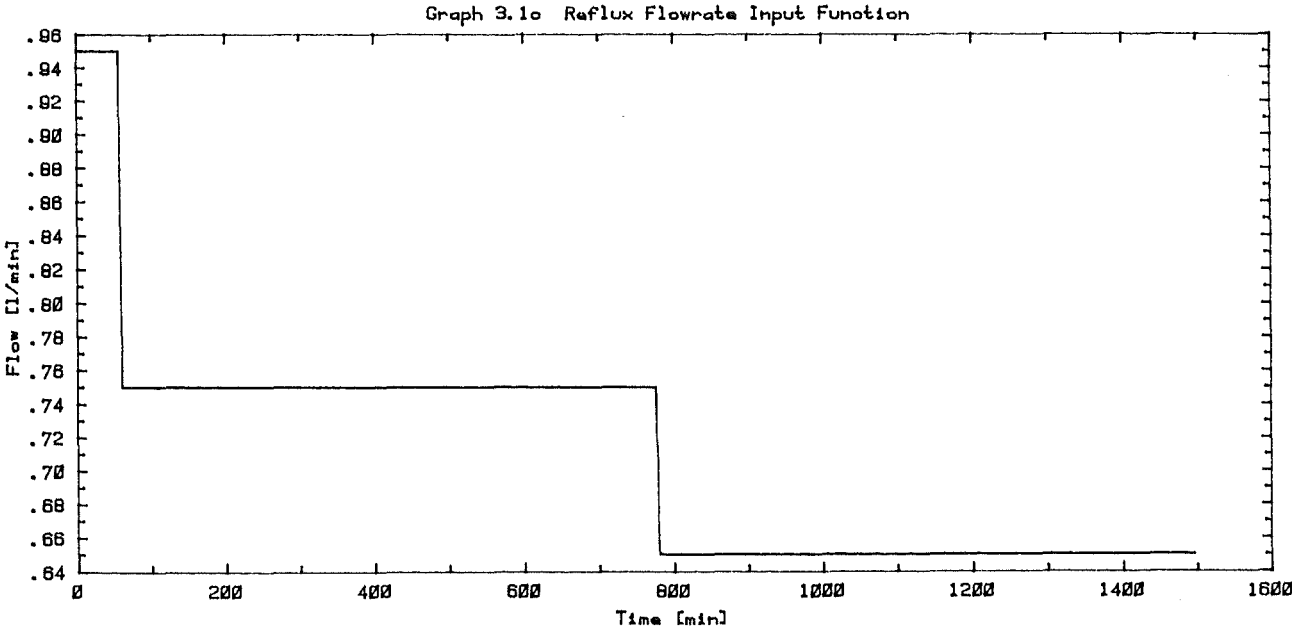
Graph 3.1a Feed Flowrate Input Function



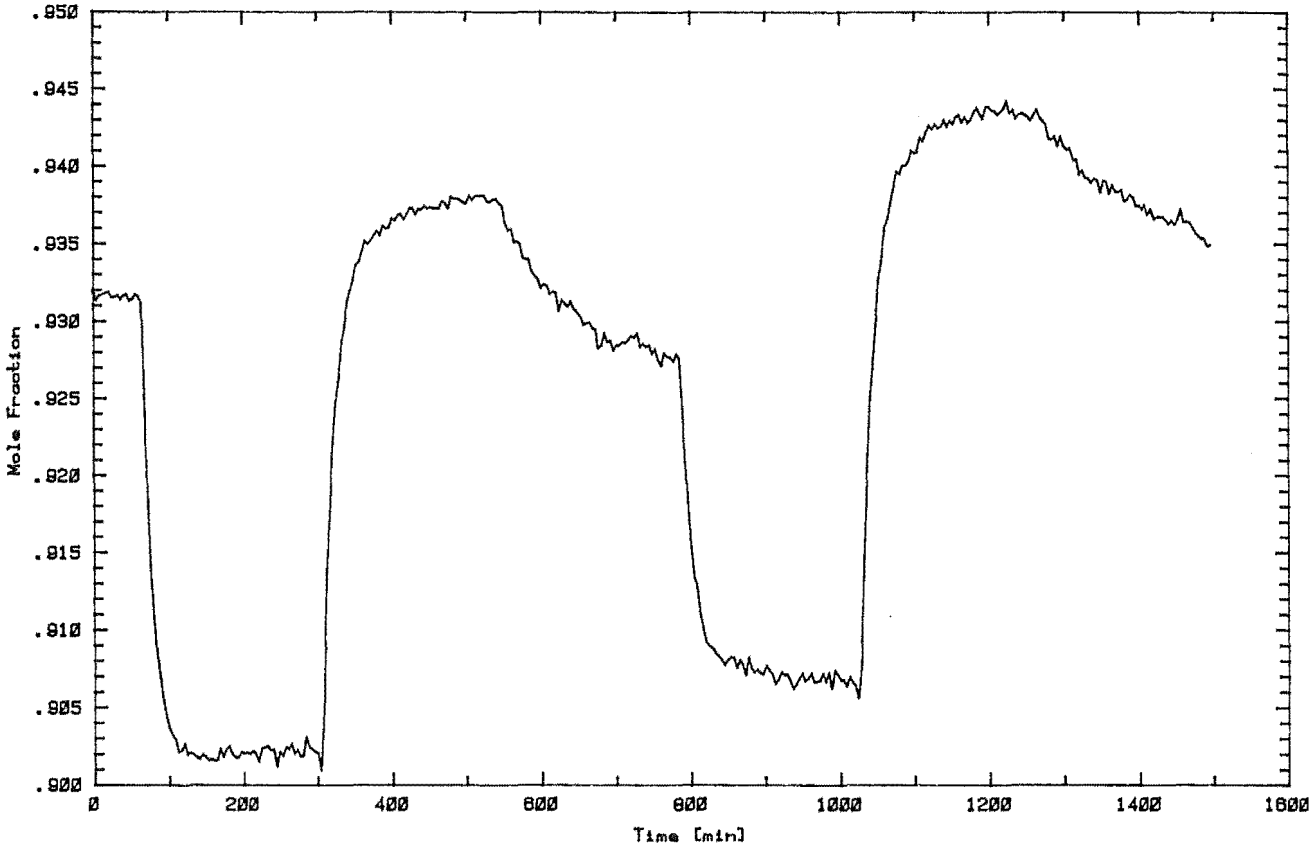
Graph 3.1b Steam Condensate Flowrate Input Function



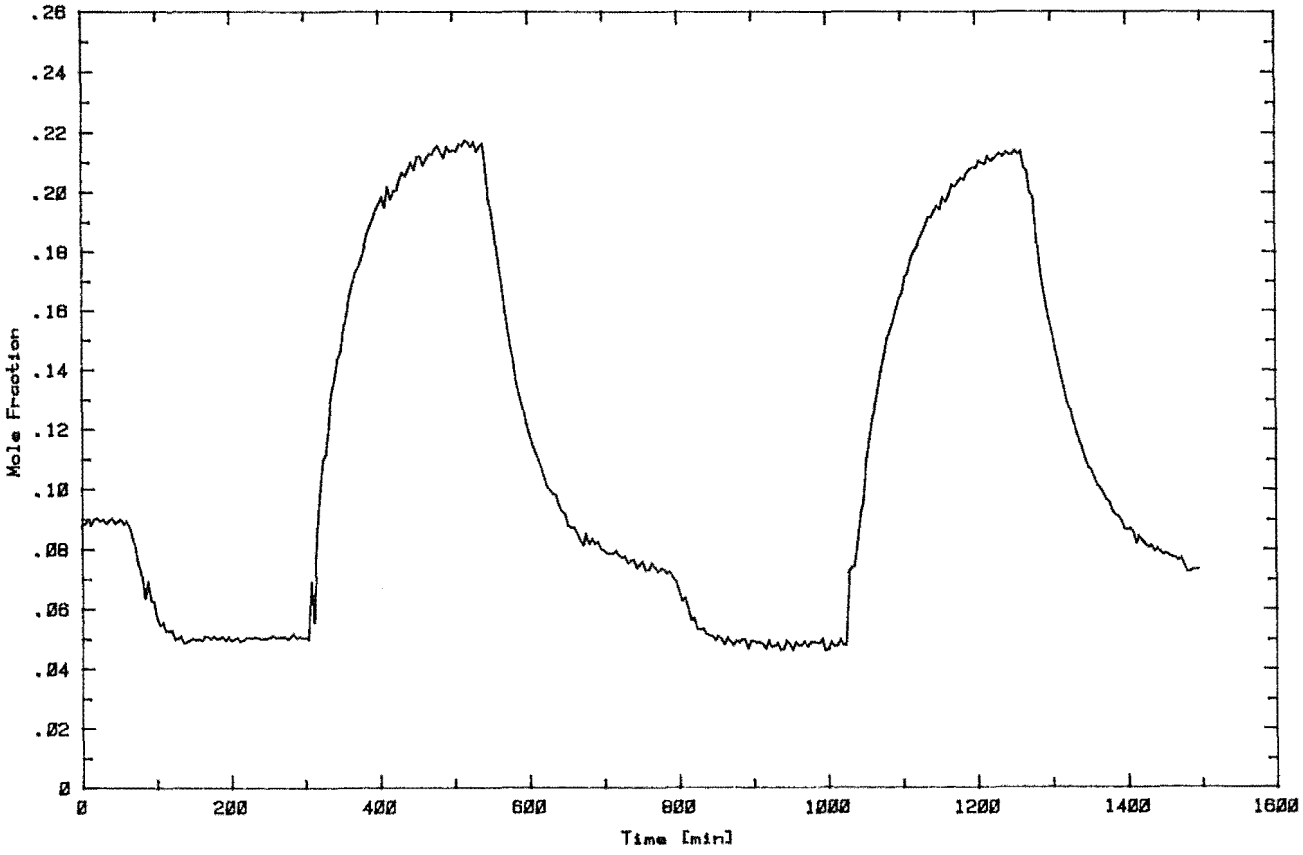
Graph 3.1c Reflux Flowrate Input Function



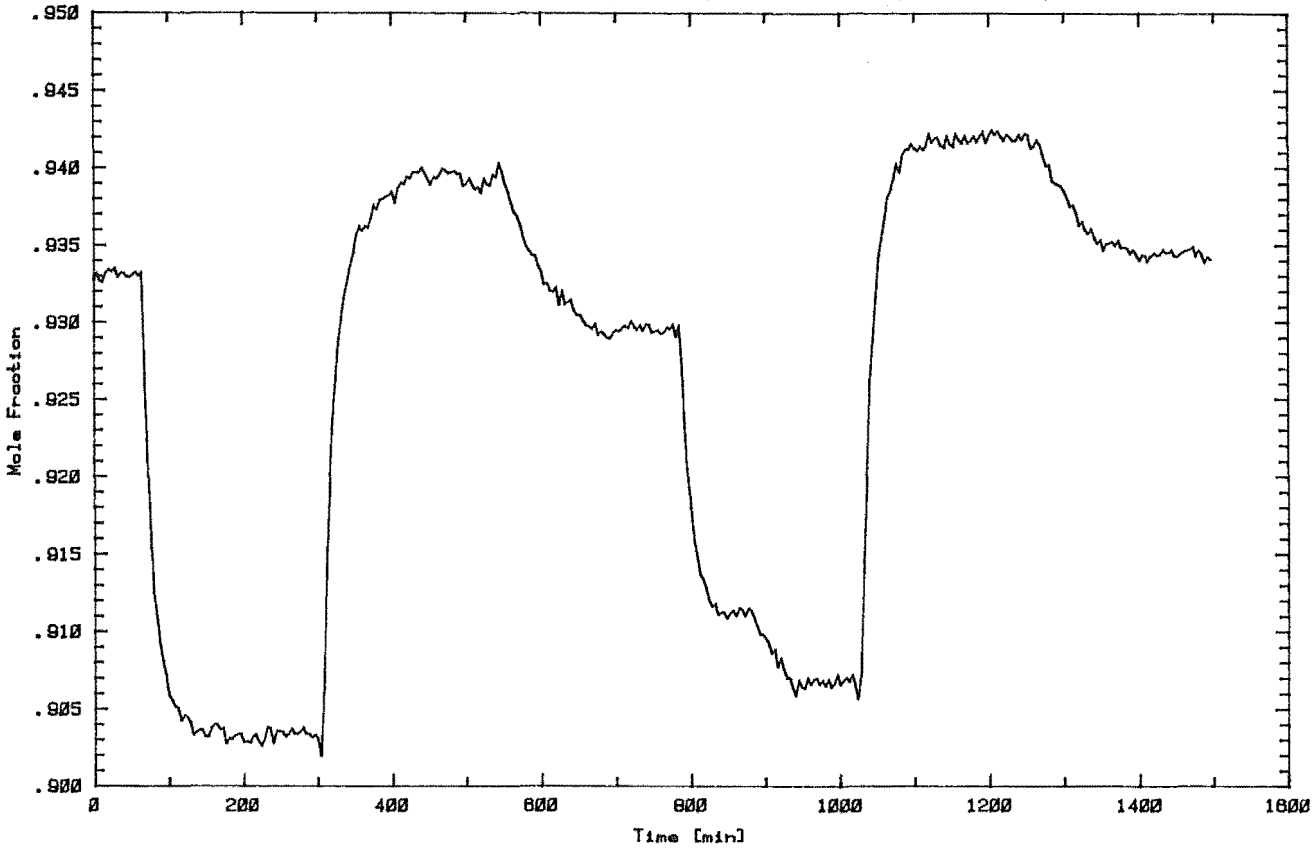
Graph 3.2a Experimental Tope Composition Reeponee



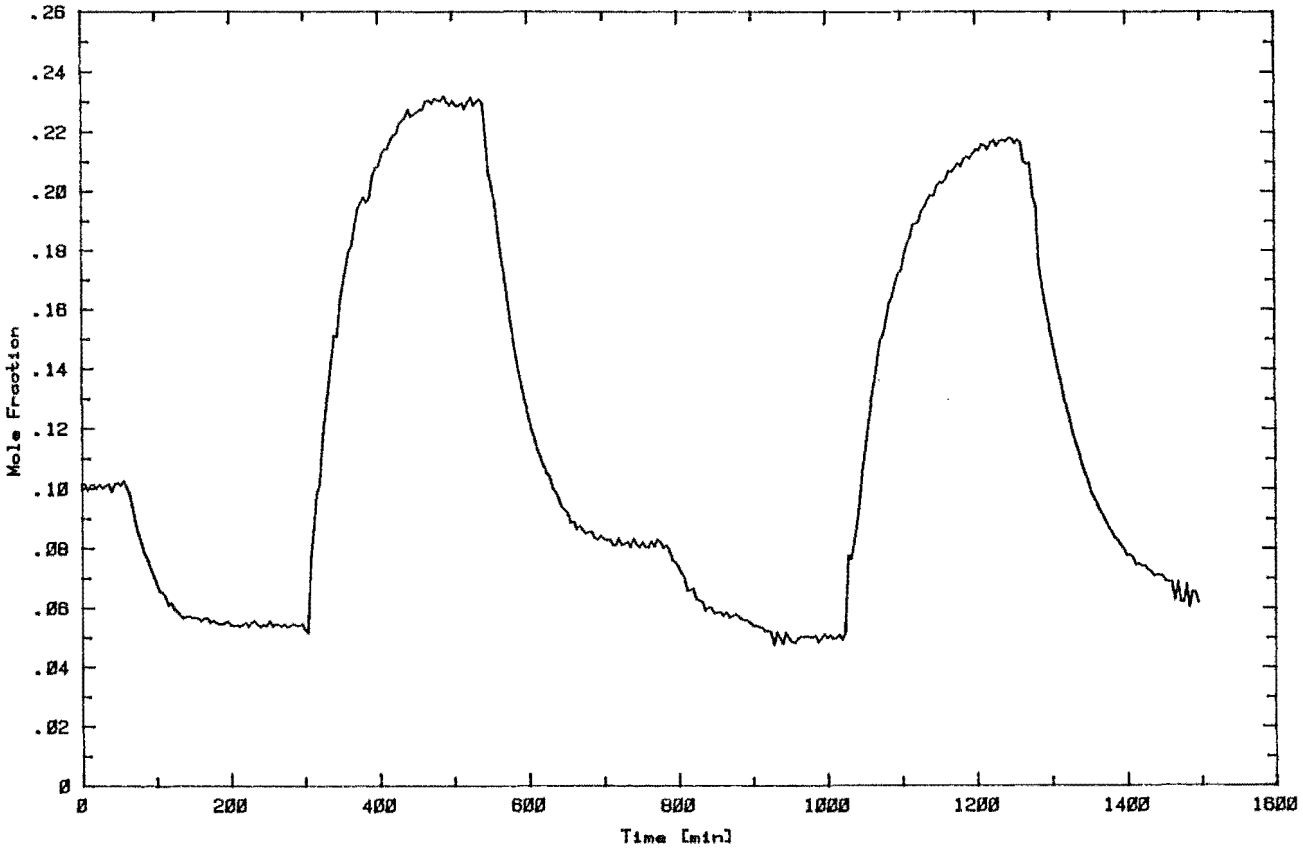
Graph 3.2b Experimental Bottoms Composition Reeponse



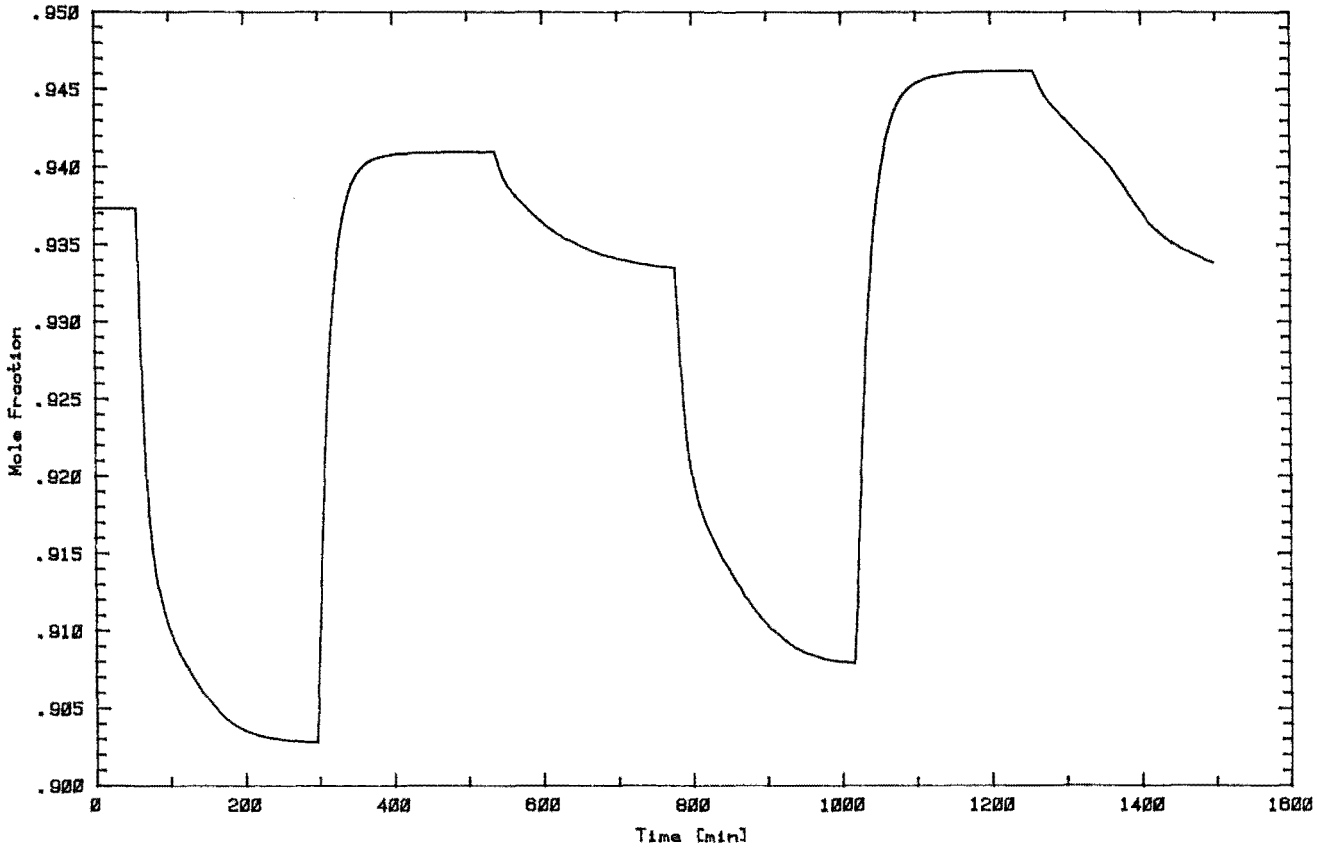
Graph 3.3a Experimental Tops Composition Response (Replication)



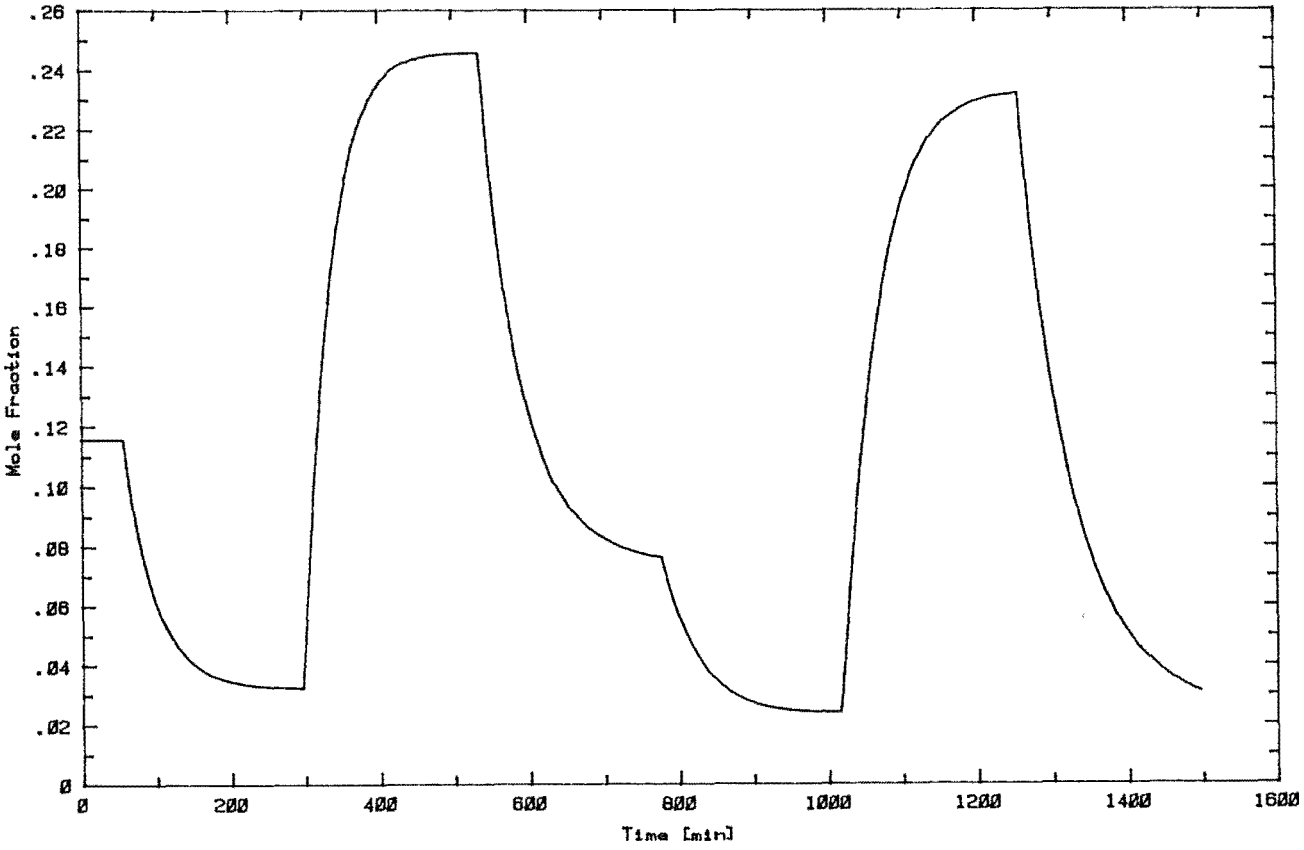
Graph 3.3b Experimental Bottoms Composition Response (Replication)



Graph 3.4a Simulated Tope Compoeition Reeponee



Graph 3.4b Simulated Bottome Composition Response



discrepancy between the simulated and experimental results was larger than the variations between the experimental replications. However, it was satisfactory for the present application because values derived from the simulation were not applied directly to the control of the experimental column, and the simulation was a closer approximation to the experimental column than the bilinear models which were identified using the simulated data.

3.9 PROGRAM PERFORMANCE

The dynamic simulation executed approximately twenty times faster on a VAX 730 mini-computer than the real process it was simulating. This performance was satisfactory for the application in this work and the programming time needed to implement an appreciably faster program was judged disproportionate to the benefits gained.

The method used in this work required the calculation of a full Jacobian matrix by finite differences, because of the difficulty in obtaining an analytic expression for the Jacobian. This meant the method was only suitable for columns with relatively small numbers of plates, because the workload increased with the square of the number of plates. Gear's method for the simultaneous solution of differential and algebraic equations results in a banded Jacobian matrix which can be calculated analytically. However, it does have a higher dimensionality due to the larger state vector. These factors make Gear's method more suitable for columns with a large number of plates.

The steady state simulation took approximately fifty seconds to converge. Although other calculation procedures may be faster, the time taken was short in comparison to that required for the dynamic simulation.

3.10 NOMENCLATURE

A_a - Active area of plate

B - Molar bottoms flowrate

C_p - Molar heat capacity

CS1 - Cross sectional area of lower section of condenser

CS2 - Cross sectional area of upper section of condenser

D - Molar distillate flowrate

E_i - Murphree vapour efficiency on the i^{th} plate

F - Molar feed flowrate

h_f - Feed molar enthalpy

h_i - Liquid molar enthalpy on the i^{th} plate

h_L - Equivalent height of clear liquid on plate

h_{ow} - Height of frothy liquid over outlet weir

h_w - Height of outlet weir

H_i - Vapour molar enthalpy on the i^{th} plate

HGT1 - Height of lower section of condenser

HGT2 - Height of upper section of condenser

k_i - Equilibrium constant on the i^{th} plate

l_w - Length of outlet weir

L_i - Liquid molar flowrate leaving the i^{th} plate

m_i - Molar holdup of the i^{th} plate

N - Number of column plates

P_c - Pressure at base of condenser

Q_p - Heat loss from a plate

Q_s - Heat input to the reboiler

Q_v - Q_s minus heat loss from the reboiler

V_i - Vapour molar flowrate leaving the i^{th} plate

x_f - Feed molar composition of methanol

x_i - Liquid molar composition of methanol on the i^{th} plate

y_i - Vapour molar composition of methanol on the i^{th} plate

y_i^* - Equilibrium vapour molar composition of methanol for the i^{th} plate

β - Aeration factor

v_c - Volume of chamber under active plate

v_r - Volume of reboiler

$\underline{\rho}$ - Molar saturated liquid density

ρ - Mass saturated liquid density

CHAPTER 4

SYSTEM IDENTIFICATION

4.1 INTRODUCTION

Identification is the determination of a mathematical model of a process or system using experimental input/output data. It consists of several distinct phases. A model type is selected taking into account *a priori* process knowledge and the application envisaged. The parameters in the chosen model are estimated using experimental input/output data. Lastly, the chosen model and estimated parameters are verified as being suitable for the specified task, and if necessary the procedure is repeated with a more suitable choice of model or parameter estimation method.

In this chapter only the aspects of identification relevant to this work are presented. Text books are available which give a more comprehensive treatment of the subject. Graupe (1972) gives an overview of a large range of classical identification techniques. Ljung and Soderstrom (1983) have developed a unified theory for recursive parameter estimation algorithms. In addition, several survey papers have appeared in the literature. Strejc (1980) deals with methods based on least squares analysis, and Gustavsson (1975) gives a summary of the application of identification techniques to chemical and physical processes.

4.2 MODEL STRUCTURE

The possible model types fall into various categories of largely different complexities. The choice of model is based on *a priori* process knowledge and on the intended application of the

model. The main model characteristics that need to be selected are :

linear	-	non-linear
continuous	-	discrete
deterministic	-	stochastic
time varying	-	time invariant
parametric	-	non-parametric

The selection of model type restricts the choice of parameter estimation techniques and may restrict the type of input function required. The use of cross correlation techniques to determine a weighting function is simplified by using an input function such as a pseudo random binary sequence, whose auto-correlation function approximates an impulse or a periodic impulse function. It may not be feasible or desirable to inject such a function, and it will cause difficulties in a multi-input multi-output system in that the input functions must be uncorrelated with each other.

Often it is possible to convert from one model type to another after its identification, such as the change from a continuous to a discrete model, which is usually achieved by a finite difference approximation. The change from a non-parametric weighting function model to a parametric difference equation can be achieved by doing a least squares fit of the impulse response functions (Isermann et al (1974)).

This work was limited to discrete models since all the data acquisition and control was done by digital computers which inherently use discrete information. The linear difference equation of order n for a single-input single-output system was used as the starting point for presenting the different techniques used in this work. It can be expressed as :

$$y(k) + a_1 y(k-1) + \ldots\ldots + a_n y(k-n) = b_1 u(k-1) + \ldots\ldots + b_n u(k-n) \quad (4.1)$$

where k is the discrete time variable, and u(k - i) and y(k - i) are the deviation variables of the system input and output at time (k - i)Δt. These are related to the actual system input and output, U(k) and Y(k), by :

$$U(k) = U_S + u(k) \quad (4.2)$$

$$Y(k) = Y_S + y(k) \quad (4.3)$$

where U_S and Y_S are the steady state input and output.

In practice, the actual system input and output are available and not the deviation variables as used in Equation 4.1. Isermann (1981) suggests a number of ways of dealing with this difficulty. The steady state input and output can be estimated by taking an average of their values over a period prior to the parameter estimation. Alternatively, both the actual inputs and outputs can be filtered with a digital high pass filter. This requires that the filter cutoff frequency be chosen so that all information about the slowest process response is preserved. The approach used in this work was to estimate an extra parameter DC and to use the actual system inputs and outputs in the expression given in Equations 4.4 and 4.5. This required no *a priori* information and worked well.

$$Y(k) + a_1 Y(k-1) + \ldots\ldots + a_n Y(k-n) = b_1 U(k-1) + \ldots\ldots + b_n U(k-n) + DC \quad (4.4)$$

where

$$DC = (1 + a_1 + \ldots + a_n) Y_S - (b_1 + \ldots + b_n) U_S \quad (4.5)$$

Dead time or pure time delay can be included in the difference model. If the dead time is an integral number of sampling intervals it results in the model :

$$Y(k) + a_1Y(k-1) + \ldots\ldots + a_nY(k-n)$$
$$= b_1U(k-d-1) + \ldots\ldots + b_nU(k-d-n) + DC \quad (4.6)$$

where the dead time is d sampling intervals long.

4.3 PARAMETER ESTIMATION

A large number of parameter estimation algorithms have been developed from a variety of perspectives. In general, the simplest algorithm that will do the task satisfactorily should be selected, because more complex algorithms are more computationally expensive, and the added generality and less *a priori* information result in a slower convergence rate. Several comparisons of different parameter estimation algorithms have appeared in the literature, the most comprehensive being those of Isermann et al (1974) and Saridis (1974).

On-line work requires the use of recursive algorithms which use each new measurement to update the estimate of the process parameters. These methods need only a small amount of computation for each measurement and do not require the storage of previous measurements. They can also be modified to follow time varying parameters. These algorithms are derived by minimising a preselected performance index which measures the discrepancy between the estimated model and the experimental process. Ljung and Soderstrom (1983) have shown that they differ only in the assumed form of the noise model and in the chosen performance criterion.

4.4 RECURSIVE LEAST SQUARES ALGORITHM

The simplest and most robust of the recursive parameter estimation techniques is the recursive least squares (RLS) algorithm. It is often used to provide initial values for other

methods because of its fast and reliable convergence. In this section the algorithm only is presented, a derivation can be found in Hsia (1977).

Let the parameter vector corresponding to Equation 4.4 be defined as :

$$\theta_o^T = [a_1, a_2 \ldots a_n, b_1 \ldots b_n, DC]$$

and the measurement vector as :

$$\varphi(k)^T = [-Y(k-1), -Y(k-2) \ldots -Y(k-n), U(k-1) \ldots U(k\ n), 1]$$

If $\theta(k)$ is the estimate of θ_o at time k then :

$$Y(k) = \varphi(k)^T\theta(k) + e(k) \tag{4.7}$$

If the performance index to be minimised is defined as :

$$J = \sum_{i=1}^{k}(Y(i) - \varphi(i)^T\theta(i))^2 \tag{4.8}$$

The result is the least squares algorithm which when given in its recursive form is :

$$\theta(k) = \theta(k-1) + \mathbf{L}(k)[Y(k) - \varphi^T(k)\theta(k-1)] \tag{4.9}$$

$$\mathbf{L}(k) = \frac{\mathbf{P}(k-1)\varphi(k)}{1 + \varphi^T(k)\mathbf{P}(k-1)\varphi(k)} \tag{4.10}$$

$$\mathbf{P}(k) = \mathbf{P}(k-1) - \frac{\mathbf{P}(k-1)\varphi(k)\varphi^T(k)\mathbf{P}(k-1)}{1 + \varphi^T(k)\mathbf{P}(k-1)\varphi(k)} \tag{4.11}$$

where $\mathbf{L}(k)$ is the gain vector and $\mathbf{P}(k)$ is the error covariance matrix.

Equation 4.9 shows that the parameter vector is corrected by a term consisting of the product of the gain vector and the output prediction error. As more measurements are taken, the estimate of the parameter vector becomes better and the size of the error covariance matrix and hence the gain vector decreases.

4.5 TIME VARYING PARAMETERS

If the identification is being done on-line in real time and the system parameters are slowly changing with time, it is possible to follow these changes by weighting the most recent measurements more heavily. The simplest method is to introduce a forgetting factor which weights the most recent results exponentially. This results in the performance index becoming :

$$J = \sum_{i=1}^{k} \lambda^{k-i} \, (Y(i)-\varphi(i)^T\theta(i))^2 \qquad (4.12)$$

where $0 < \lambda \leq 1$ this results in Equations 4.10 to 4.11 becoming:

$$\mathbf{L}(k) = \frac{\mathbf{P}(k-1)\varphi(k)}{\lambda + \varphi^T(k)\mathbf{P}(k-1)\varphi(k)} \qquad (4.13)$$

$$\mathbf{P}(k) = \lambda\mathbf{P}(k-1) - \frac{\lambda\mathbf{P}(k-1)\varphi(k)\varphi^T(k)\mathbf{P}(k-1)}{\lambda + \varphi^T(k)\mathbf{P}(k-1)\varphi(k)} \qquad (4.14)$$

This change prevents the error covariance matrix and hence the gain vector from tending to zero as they would otherwise do. There are practical limits on the sizes that can be chosen for the forgetting factor λ. If it is chosen too small, the estimated parameters fluctuate excessively with measurement noise. If chosen too close to unity, the speed of adaption will be too slow. In practice, it is chosen in the range 0.95 to 0.99 (Isermann 1981).

4.6 CORRELATED NOISE

It has been shown (Hsia (1977)) that if measurement noise is present the residual in the RLS algorithm is correlated with the measurements, resulting in the estimated parameters converging, but not to the correct values. The results are biased. This problem may be overcome by extending the algorithm to identify the noise parameters as well as the system parameters. This results in algorithms such as the extended least squares, the instrumental variable technique and the generalised least squares. The simplest of these is the extended least squares where the model is assumed to be of the form :

$$\begin{aligned} Y(k) &+ a_1Y(k-1) + \ldots\ldots + a_nY(k-n) \\ &= b_1U(k-1) + \ldots\ldots + b_nU(k-n) \\ &+ c_1e(k-1) + \ldots\ldots + c_ne(k-n) + DC \end{aligned} \qquad (4.15)$$

The parameter vector given in Section 4.4, is now augmented with the noise parameters and the residuals are fed back as estimates of the output measurement noise giving the following expressions :

$$\theta_o^T = [a_1,\ a_2 \ldots a_n,\ b_1 \ldots b_n,\ c_1 \ldots c_n,\ DC]$$

$$\varphi(k)^T = [-Y(k-1),\ -Y(k-2) \ldots -Y(k-n),\ U(k-1) \ldots U(k-n),$$
$$e(k-1) \ldots e(k-n),\ 1]$$

The overall effect of these algorithms is that they adaptively filter the measurements before using them to estimate the parameters. This demonstrates the connection between these methods and the extended Kalman filter, the latter being a filtering algorithm modified to simultaneously estimate uncertain parameters.

4.7 U-D FACTORISATION ALGORITHM

The recursive least squares algorithm sometimes converges poorly or results in the manipulation of ill-conditioned matrices. It has been shown (Bierman (1977)) to go unstable if at any stage the error covariance matrix **P** becomes negative definite. The square root estimation algorithm and the U-D factorisation algorithm overcome this difficulty by updating a factor of **P** such that **P** stays positive definite, rather than updating **P** itself.

In this section, the U-D factorisation algorithm due to Bierman (1977) is presented. This algorithm factorises the matrix **P** into $\mathbf{UDU}^T$, where **D** is a diagonal matrix and **U** is an upper triangular matrix with a unit diagonal. This is done using the numerically efficient Cholesky decomposition. The U-D factorisation algorithm typically gives answers of the same accuracy, using only half the precision of arithmetic as the RLS algorithm presented previously. Unlike the square root algorithm, it does not require the calculation of any square roots.

Let θ and φ be the parameter and measurement vectors of dimension m as defined in the RLS algorithm. **f**, **g** and $\mathbf{k}_i$ are m dimensional vectors used to store intermediate values, as are the α_i scalars. **U** and **D** are m x m dimension arrays.

$$\mathbf{f}^T = [f_1,\ f_2\ \ldots\ f_m]$$

$$\mathbf{g}^T = [g_1,\ g_2\ \cdots\ g_m]$$

$$\mathbf{D} = \begin{bmatrix} d_1(k) & 0 & \ldots\ldots & 0 \\ 0 & d_2(k) & 0\ \ldots & 0 \\ - & - & - & - \\ 0 & \ldots\ldots & 0 & d_m(k) \end{bmatrix}$$

$$\mathbf{U} = [\mathbf{u}_1(k),\ \mathbf{u}_2(k)\ \ldots\ \mathbf{u}_m(k)]$$

The following procedure is then repeated for every measurement :

1. $\mathbf{f} = \mathbf{U}^T(k-1)\mathbf{x}(k)$
2. $\mathbf{g} = \mathbf{D}(k-1)\mathbf{f}$
3. $\alpha_1 = \lambda + g_1 f_1$
4. $d_1(k) = d_1(k-1)/\alpha_1$
5. $\mathbf{l}_2^T = (g_1, 0 \ldots 0)$
6. Repeat steps 7 to 10 for j = 2 to m
7. $\alpha_j = \alpha_{j-1} + g_j f_j$
8. $d_j(k) = d_j(k-1)\alpha_{j-1}/\alpha_j\lambda$
9. $\mathbf{u}_j(k) = \mathbf{u}_j(k-1) + \mu_j \mathbf{l}_j$ where $\mu_j = -f_j/\alpha_{j-1}$
10. $\mathbf{l}_{j+1} = \mathbf{l}_j + g_j \mathbf{u}_j(k-1)$

Finally the gain $\mathbf{L}(k+1)$ is given by:

$$\mathbf{L}(k) = \mathbf{l}_{m+1}/\alpha_m \tag{4.16}$$

and the parameter estimates are updated by

$$\theta(k) = \theta(k-1) + \mathbf{L}(k)[Y(k) - \varphi^T(k)\theta(k-1)] \tag{4.17}$$

It is more efficient not to calculate the gain L(k) directly, but to update the parameters using the normalised gain $\mathbf{l}_{m+1}$ and the innovations variance α_m using the expressions :

$$\varepsilon = [Y(k) - \varphi^T(k)\theta(k-1)]/\alpha_m \tag{4.18}$$

$$\theta(k) = \theta(k-1) + \mathbf{l}_{m+1}\varepsilon \tag{4.19}$$

At the start of the identification, $\mathbf{U}$ and θ are set to zero, d_i's are initialised to a large value typically 10^4, and λ is in the range 0.95 to 1.0 .

4.8 NON-LINEAR PARAMETER ESTIMATION

There are two types of non-linear equation, those in which the parameters to be identified appear linearly and those in which they appear non-linearly. If the parameters appear linearly, they can be estimated by a straightforward application of the recursive

least squares algorithm. The extensions to deal with correlated noise, as shown in Section 4.6, are more difficult. As an example, the following non-linear equation is linear in the parameters :

$$Y(k) = aY(k-1) + bY^2(k-1) + cV(k-1) + dY(k-1)V(k-1) + DC \qquad (4.20)$$

It can be identified using the RLS algorithm with the parameter and measurement vectors defined as :

$$\theta_o^T = [a,\ b,\ c,\ d,\ DC]$$

$$\varphi(k)^T = [Y(k-1),\ Y^2(k-1),\ V(k-1),\ Y(k-1)V(k-1),\ 1]$$

The parameters are considerably more difficult to estimate if they appear non-linearly. One approach, which can be applied in most situations, is to use what are sometimes termed heuristic techniques. In these methods, a performance criteron is chosen and a minimisation procedure is used to find the parameters which give the best index. This approach is not recursive and multi-dimensional optimisation procedures are computationally expensive.

4.9 INPUT SIGNALS

The system must be in a transient state and persistently excited to the required order, so as to identify dynamic parameters. A sine wave is normally inadequate because it contains information at only one frequency. Similarly, to identify a non-linear model, the measured inputs and disturbances must be such that the non-linearities are excited. This will usually require that the operating data cover the whole range over which the model is to be valid.

Some parameter estimation techniques can in theory use normal operating data, but there are very few examples in the literature where their application has been successful. A high signal to

noise ratio is necessary for a feasible rate of convergence. This can be achieved by making the input signals as large as possible and by adequate analogue filtering of the measurements. For linear models, the input signal size is constrained by the need to avoid exciting non-linearities, but for non-linear models the opposite is true.

If parameter estimation techniques are being used that assume a deterministic model or have a simplified noise model, it is necessary to have a high signal to noise ratio to minimise the bias of the estimated parameters.

4.10 SAMPLING RATE

In the deterministic case, with no process or measurement noise, no discretisation error and known dead time, the sampling time can be chosen to be very short. In the real situation, the possible sample time is bounded at the lower end by factors such as noise and discretisation error, and at the upper end by the control performance. Isermann (1981) suggests using a sampling time in the range :

$$\frac{1}{15} T_{95} < \Delta t < \frac{1}{4} T_{95}$$

where T_{95} is the time for the process to reach 95% of its final value after a step change in one of the inputs. Gustavsson (1975) suggests a sampling time of one tenth of the major system time constant.

The choice appears not to be critical. The sampling theorem states that the sampling frequency must be at least twice the frequency of the fastest component of interest. In practice, the sampling frequency must be several times faster than the Nyquist frequency. Gustavsson (1975) suggests a conservative choice of a sampling time equal to the shortest time

constant of interest. This is equivalent to sampling at about six times the frequency of the fastest component.

One of the problems in sampled data systems is that frequency components that are faster than half the sampling frequency appear superimposed on lower frequencies in what is known as aliasing. The measurements should be filtered with analogue filters so as to remove these high frequencies.

The length required by an identification experiment is determined by the convergence rate which is affected by the choice of model and the amount of noise present. In practice, a compromise has to be made between rate of convergence and model complexity. Gustavsson (1975) suggests that, as a guide, the experiment length be at least ten times as long as the major time constant of the system.

4.11 VERIFICATION

The ultimate verification of the identified model is whether it is satisfactory for its intended purpose which is usually control of the process. However, it is often necessary to be able to verify the model independently of the control operation.

The simplest verification procedure is a visual comparison of the outputs of the experimental process and the estimated model for the input sequence used in the identification. Alternatively, a visual comparison can be made for a different sequence. This tests the ability of the identified model to predict process behaviour as well as verifying the quality of fit.

A less subjective test is the evaluation of a performance index for the above mentioned sequences. An obvious choice is the variance between the experimental process and the estimated model output sequences.

For a simulated process, it is often possible to extract composite parameters such as gains and time constants from the

identified model and to compare these directly with the expected theoretical parameters.

The size of the prediction error, especially towards the end of the identification, is a measure of modelling accuracy if little noise is present. This can also be used in on-line identification as an indicator of the quality of fit and hence, Fortescue et al (1981) have used it in their calculation of a variable forgetting factor.

CHAPTER 5

BILINEAR SYSTEMS

5.1 INTRODUCTION

It is often assumed for the identification and control of a process that it behaves in a linear manner about some operating point and can be adequately described by a linear state space model of the form :

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + B\mathbf{u} \tag{5.1}$$

$$\mathbf{y} = C\mathbf{x} \tag{5.2}$$

where $\mathbf{x}$ is an n dimensional state vector, $\mathbf{u}$ is a p dimensional input vector, $\mathbf{y}$ is an m dimensional measurement vector, and A,B and C are constant matrices of appropriate dimensions.

Linear systems are attractive because their theoretical aspects are now well developed, and because the identification and control of linear systems have convenient analytical solutions (Kailath 1980).

If a process is strongly non-linear or operates over a wide operating range the linear assumption may be invalid. One solution is to use a non-linear model, but it is not feasible to use a completely general non-linear model due to the complexity and computational effort required to solve the identification and control problem. Bilinear systems are a class of non-linear system that has attracted much attention. Bilinear systems are linear in both the states and the inputs when considered separately, but not when considered jointly. They can be

described by the following time invariant continuous state space representation (Mohler 1973) :

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + \sum_{i=1}^{p} u_i N_i \mathbf{x} + B\mathbf{u} \tag{5.3}$$

$$\mathbf{y} = C\mathbf{x} \tag{5.4}$$

where N_i's are n x n constant matrices and the other variables are as described above. If all the elements of the N_i matrices are zero it is clear that linear systems are a subset of bilinear systems.

The continuous state space representation given above can be converted to its discrete counterpart by a finite difference approximation :

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + \sum_{i=1}^{p} u_i(k) N_i \mathbf{x}(k) + B\mathbf{u}(k) \tag{5.5}$$

$$\mathbf{y}(k) = C\mathbf{x}(k) \tag{5.6}$$

Considerable theoretical and experimental work has been done on the identification and control of bilinear systems, but it is still far from being complete.

Initial interest in bilinear systems came as the result of work into the modelling of growth functions in biological systems and chain reactions in nuclear systems (Mohler 1973).

The paper of Bruni et al (1974) presented a summary of the field as it stood at that time. Results for determining the observability and controllability of bilinear systems were presented and a canonical decomposition developed.

D'Allesandro et al (1974) develop realisation theory based on Volterra series. They provide a procedure to determine if a particular realisation is a minimal one.

The work of Svoronos et al (1980) presents two methods to obtain a bilinear approximation from the general non-linear system equations. They show that the advantage of a bilinear approximation over a linear one can be related to the retention of higher order terms in the truncation of the Taylor series expansion. The more terms that are included the more accurate the bilinear approximation, but at the cost of a greater model dimensionality.

The situation often arises in chemical engineering where a plant has a fixed volume and the flow through the plant is the control variable. The equations derived by performing a mass or energy balance around such a plant result in a model of the bilinear form. It has been shown by Espana and Landau (1975) that the mass balance equations for a distillation column can be reduced to this form if certain restrictive assumptions are made.

5.2 OBSERVABILITY

The observability of bilinear systems has been studied by Williamson (1977) and Funahashi (1979). This analysis is based on considering the bilinear system as a time varying linear system in which the parameters vary in a forced manner.

$$\mathbf{x}(k+1) = L(k)\mathbf{x}(k) + B\mathbf{u}(k) \tag{5.7}$$

$$L(k) = A + \sum_{i=1}^{p} u_i(k) N_i \tag{5.8}$$

$$\mathbf{y}(k) = C\mathbf{x}(k) \tag{5.9}$$

The observability matrix for this system can be defined as shown in Equation 5.10. For linear systems all the elements of **T**(k) are constant but for bilinear systems they are polynomials in u(k),u(k-1) ... u(k+1-n).

$$\mathbf{T}(k) = \begin{bmatrix} C \\ CL(k) \\ CL(k+1)L(k) \\ \hline CL(k+n-2)\ldots L(k) \end{bmatrix} \qquad (5.10)$$

The system is uniformly observable if and only if the matrix **T**(k) has rank of n.

Goodwin and Sin (1984) have defined systems for which the matrix **T**(k) has a constant determinant as strongly uniformly observable.

5.3 BILINEAR DIFFERENCE EQUATIONS

It is preferable to have the system described by a discrete difference type of equation for identification purposes. Two methods appear in the literature for transforming the discrete state space model into a difference equation. Goodwin and Sin (1984) use a method based on successive substitution. Beghelli and Guidorzi (1976) transform the model into row companion form, and hence to a difference equation form, in a manner analogous to that used for linear systems. Both methods require that the

bilinear system is observable and result in a difference equation of the form :

$$y(k+1) = \sum_{i=1}^{n} \beta_i y(k+1-i) + \beta_0 \qquad (5.11)$$

where the βi's are non-linear functions of $u(k), u(k-1) \ldots u(k+1-n)$.

There are a large number of terms in the full difference equation, Beghelli and Guidorzi (1976) list no fewer than 88 terms for a third order bilinear system. This has the problem that high order or multi-input multi-output systems become unmanageable and if there are too many terms with similar effects on the I/O map, there is the possiblity of getting sets of parameters that are almost dependent. These parameters will contribute very little to the closeness of fit, but will slow the rate of convergence of the identification algorithm. These difficulties encouraged several researchers (Beghelli and Guidorzi (1976), Subba Rao and Gabr (1984)) to develop identification procedures which evaluated only the most significant parameters in the full bilinear model.

An alternative approach is to use a subset of the full bilinear system. For strongly observable bilinear systems the non-linear β_i functions are polynomials in $u(k), u(k-1) \ldots u(k+1-n)$, but the reduction in terms and complexity is still not large.

If the non-linear β_i functions are approximated by linear functions in $u(k), u(k-1) \ldots u(k+1-n)$ then a model as shown in Equation 5.12 results. Several researchers (Svoronos et al (1981), Subba Rao and Gabr (1984)) have used this formulation, and the justification for its use is that the approximation is still an order of approximation better than a purely linear system approximation.

$$
\begin{aligned}
y(k) &+ a_1 y(k-1) + \ldots + a_n y(k-n) \\
&= b_1 u(k-1) + \ldots + b_n u(k-n) \\
&+ c_1{}^1 y(k-1)u(k-1) + c_1{}^2 y(k-1)u(k-2) + \ldots + c_1{}^n y(k-1)u(k-n) \\
&+ c_2{}^1 y(k-2)u(k-1) + c_2{}^2 y(k-2)u(k-2) + \ldots + c_2{}^n y(k-2)u(k-n) \\
&+ \ldots\ldots \\
&+ c_n{}^1 y(k-n)u(k-1) + \ldots + c_n{}^n y(k-n)u(k-n) \qquad (5.12)
\end{aligned}
$$

where $y(k-i)$ and $u(k-i)$ are the deviation variables of output and input at time $(k-i)\Delta t$.

In the subset bilinear difference equation given in equation 5.12 the number of parameters still increases with n^2 rather than with n as in linear systems. Ohkawa and Yonezawa (1983) only include the multiplicative terms with the coefficients $C_1{}^1$, $C_2{}^2$ ... $C_n{}^n$. Subba Rao and Gabr (1984) have termed this subset a diagonal bilinear system and it has the advantage that the number of parameters only increases proportionally to n.

Equation 5.12 is given in terms of deviation variables rather than measured signals. It can be converted to measured signals by substituting the Equations 5.13, 5.14 and 5.15 into equation 5.12. The result is the addition of a DC term as shown in Equation 5.16, but in the conversion, the a_i and b_i coefficients are changed, unlike the linear situation. The new values of the coefficients and the DC term are given in Equations 5.17, 5.18 and 5.19.

$$U(k-i) = U_S + u(k-i) \qquad (5.13)$$

$$Y(k-i) = Y_S + y(k-i) \qquad (5.14)$$

$$
\begin{aligned}
U(k-i)Y(k-j) &= [U_S + u(k-i)] * [Y_S + y(k-j)] \\
&= U_S Y_S + Y_S u(k-i) + U_S y(k-j) + u(k-i)y(k-j) \qquad (5.15)
\end{aligned}
$$

$$
\begin{aligned}
&Y(k) + \underline{a}_1 Y(k-1) + \ldots + \underline{a}_n Y(k-n) \\
&\quad = \underline{b}_1 U(k-1) + \ldots + \underline{b}_n U(k-n) \\
&\quad + c_1{}^1 Y(k-1)U(k-1) + c_1{}^2 Y(k-1)U(k-2) + \ldots + c_1{}^n Y(k-1)U(k-n) \\
&\quad + c_2{}^1 Y(k-2)U(k-1) + c_2{}^2 Y(k-2)U(k-2) + \ldots + c_2{}^n Y(k-2)U(k-n) \\
&\quad + \ldots\ldots \\
&\quad + c_n{}^1 Y(k-n)U(k-1) + \ldots + c_n{}^n Y(k-n)U(k-n) + DC
\end{aligned}
\tag{5.16}
$$

$$
\underline{a}_i = a_i + U_s \sum_{j=1}^{n} c_i{}^j \tag{5.17}
$$

$$
\underline{b}_i = b_i - Y_s \sum_{i=1}^{n} c_i{}^j \tag{5.18}
$$

$$
DC = U_s Y_s \sum_{i=1}^{n} \sum_{j=1}^{n} c_i{}^j - U_s \sum_{i=1}^{n} b_i + Y_s \left(1 + \sum_{i=1}^{n} a_i\right) \tag{5.19}
$$

Equation 5.12 can be linearised about an operating point by approximating the u(t-i)y(t-j) term to zero. The process gain K_p of y with respect to u can be expressed as :

$$
K_p = \frac{\sum_{i=1}^{n} b_i}{1 + \sum_{i=1}^{n} a_i} \tag{5.20}
$$

If the model coefficients have been found in terms of measured signals then the process gain of the bilinear system at the point U_s:Y_s can be found by :

$$K_p = \frac{\sum_{i=1}^{n} b_i + Y_s \sum_{i=1}^{n}\sum_{j=1}^{n} c_i{}^j}{1 + \sum_{i=1}^{n} a_i - U_s \sum_{i=1}^{n}\sum_{j=1}^{n} c_i{}^j} \qquad (5.21)$$

Linear multi-input multi-output (MIMO) systems have the desirable property that they can be subdivided into multiple MISO linear systems. Mathematically this can be demonstrated by transforming the system to its p-canonical form (Isermann 1981). MIMO bilinear systems can also be subdivided into multiple MISO systems. This is shown by the fact that each member in the output vector **y** is only a function of the previous input vectors and the initial system state, but not of the other system outputs.

5.4 IDENTIFICATION

5.4.1 PARAMETER ESTIMATION ALGORITHMS

Bilinear systems have the major advantage that they are linear in the parameters. This means that many of the identification techniques used for linear systems are also applicable to bilinear systems (Hsia (1977)) and the classical recursive least squares algorithm falls in this category. The algorithms given in Chapter 4 can be applied directly to bilinear systems if the parameter and measurement vectors are defined as :

$$\theta_0{}^T = [a_1, a_2 \ldots a_n, b_1 \ldots b_n, c_1{}^1, c_1{}^2 \ldots c_n{}^n, DC]$$

$$\varphi(k)^T = [-Y(k-1), -Y(k-2) \ldots -Y(k-n), U(k-1) \ldots$$
$$U(k-n), Y(k-1)U(k-1) \ldots Y(k-n)U(k-n), 1]$$

This was the approach used in this work and was similar to that used by Goodwin and Sin (1984). It has a number of

advantages. There is no limitation on the form of the input function except that it must be persistently exciting. If necessary, the identification can be performed in real-time to track time-varying parameters.

As with linear systems, the algorithm will converge to the true parameters in the deterministic case (Svoronos et al (1981)), but in a stochastic environment it will yield biased results. In this work, these difficulties were avoided by minimising sources of noise and by choosing the input signals to provide a reasonable signal to noise ratio.

A number of researchers have modified the stochastic approximation algorithm of Saridis and Stein (1968) for use with bilinear systems. Inagaki (1982) has dealt with the situation of a known input function and Kubrusly (1981) with an unmeasured random input signal of known distribution. The stochastic approximation algorithm makes few assumptions on the nature of the process to be identified, and as a result, the variances of the parameter estimates are large for small measuring periods (Isermann (1981)).

A couple of novel identification techniques have been suggested for bilinear systems. Elhennawey et al (1982) used an identification technique based on model reference adaptive system techniques to identify the parameters of a simulated bilinear system. Rao et al (1978) presented an identification method based on approximating the input and output functions by their Walsh function expansions. This method was then applied to two simulated bilinear systems.

Espana and Landau (1978) estimated the parameters in a continuous state space model of a distillation column. They used a quasi-Newton technique to minimise a performance criterion consisting of the sum of squares of the output errors. This is the same criterion as that used by the recursive least squares

algorithm above and hence will have the same deficiencies. It has the additional disadvantage that it is a batch technique and can only be used off-line.

5.4.2 INPUT FUNCTION SELECTION

A necessary requirement for the input signal is that it is persistently exciting or rich enough to activate all the modes of the process to be identified. In this section, the conditions are derived for a signal to persistently excite a bilinear process. The derivation is similar to that used by Isermann (1980) for linear systems.

For an n^{th} order bilinear system, let the parameter and measurement vectors be defined as follows :

$$\theta_o^T = [a_1,\ a_2 \ \ldots\ a_n,\ c_1^1,\ c_1^2 \ \ldots\ c_n^n,\ b_1 \ \ldots\ b_n] \tag{5.22}$$

$$\begin{aligned}\varphi(k)^T = [&-y(k-1), -y(k-2) \ \ldots -y(k-n),\ y(k-1)u(k-1),\\ &y(k-1)u(k-2) \ \ldots\ y(k-n)u(k-n),\\ &u(k-1),\ u(k-2) \ \ldots\ u(k-n)]\end{aligned} \tag{5.23}$$

Let Ψ be a (N-n) x (n^2+2n) matrix of measurements, where N is the number of measurements. Let $\mathbf{y}$ be an (N-n) dimensional vector containing the measured outputs.

$$\Psi = \begin{bmatrix} \varphi(n+1) \\ \vdots \\ \varphi(N) \end{bmatrix} \tag{5.24}$$

$$\mathbf{y}^T = [y(n+1),\ \ldots . y(N)] \tag{5.25}$$

For non-recursive least squares identification, an estimate of the parameter vector is found by :

$$\theta = [\Psi^T\Psi]^{-1}\Psi^T\mathbf{y} \tag{5.26}$$

The matrix $\Psi^T\Psi$ must be nonsingular for a solution to exist. The same condition holds for the matrix H, defined as :

$$H = \frac{1}{N} \Psi^T\Psi \tag{5.27}$$

If the number of measurements N is large, then matrix H tends to the following quadratic and symmetric form :

$$\begin{bmatrix} \phi_{yy}(0)\ldots\phi_{yy}(n-1) & : -\phi_{yyu}(0,0)\ldots-\phi_{yyu}(n-1,n-1) & : -\phi_{yu}(0)\ldots-\phi_{yu}(n-1) \\ \phi_{yy}(0) & : -\phi_{yyu}(n-1,n-1)\ldots-\phi_{yyu}(0,0) & : -\phi_{yu}(n-1)\ldots-\phi_{yu}(0) \\ \ldots & \ldots & \ldots \\ & : \phi_{yyuu}(0,0,0)\ldots\phi_{yyuu}(n-1,0,n-1) & : \phi_{yuu}(0,0)\ldots\phi_{yuu}(0,n-1) \\ & : \phi_{yyuu}(0,0,0) & : \phi_{yuu}(0,n-1)\ldots\phi_{yuu}(0,0) \\ \ldots & \ldots & \ldots \\ & : & : \phi_{uu}(0)\ldots\phi_{uu}(n-1) \\ & : & : \phi_{uu}(0) \end{bmatrix}$$

(5.28)

where the correlation functions are defined as follows :

$$\phi_{yy}(i) = \sum_{k=1}^{N} y(k)y(k+i) \tag{5.29}$$

$$\phi_{uu}(i) = \sum_{k=1}^{N} u(k)u(k+i) \tag{5.30}$$

$$\phi_{yu}(i) = \sum_{k=1}^{N} y(k)u(k+i) \tag{5.31}$$

$$\phi_{yuu}(i,j) = \sum_{k=1}^{N} y(k)u(k+i)u(k+j) \tag{5.32}$$

$$\phi_{yyu}(i,j) = \sum_{k=1}^{N} y(k)y(k+i)u(k+j) \tag{5.33}$$

$$\phi_{yyuu}(i,j,l) = \sum_{k=1}^{N} y(k)y(k+i)u(k+j)u(k+l) \tag{5.34}$$

If the matrix H is partitioned as shown below, the partition H_{33} is the only one that depends solely on the input function.

$$H = \begin{bmatrix} H_{11} : H_{12} : H_{13} \\ \cdots\cdots\cdots\cdots \\ H_{21} : H_{22} : H_{23} \\ \cdots\cdots\cdots\cdots \\ H_{31} : H_{32} : H_{33} \end{bmatrix} \tag{5.35}$$

The determinant of H must not be equal to zero for a solution to exist. Isermann (1980) has shown that the first two principal minors of H, starting from the lower righthand corner, are positive. This means that H must be positive definite for a solution to exist, and all principal minors, including that for H_{33}, must be positive. Hence a necessary, but not sufficient, condition for the solution to Equation 5.26 to exist is that the input function be chosen such that:

$$\det H_{33} > 0 \tag{5.36}$$

where H_{33} is evaluated as follows :

$$H_{33} = \{h_{ij} = \phi_{uu}(i-j)\} \qquad i,j = 1,2 \ \ldots \ ,n \tag{5.37}$$

This condition has been termed "persistent excitation of order n", and is similar to that required for linear systems. In this work, a random binary sequence was used as an input function. This type of sequence has an impulse autocorrelation function, which results in H_{33} being a diagonal matrix with

positive elements along the diagonal, and hence, the persistent excitation criterion is satisfied.

In addition to being persistently exciting, the input signal should also reflect the normal operating data (Ljung and Soderstrom (1983)). In this work, the random binary sequence has been superimposed on a three levelled step function, in order to drive the system over a wide operating region.

A number of researchers have looked at the design of input functions that minimise some form of performance index (Mehra (1981), Zarrop (1979)). Mehra (1981) presents a method which could be applied directly to bilinear systems and is based on using a direct search technique to maximise the determinant of the Fisher information matrix for nonlinear systems. In general, the use of these optimal design techniques is not warranted unless experiments are expensive and the inherent assumptions are valid (Isermann (1980)), and hence, this approach was not pursued in this work.

5.5 CONTROL

This work is mainly concerned with the identification of bilinear systems, but the eventual application of an identified model would almost certainly be in the control field. This section contains a brief summary of the state of the art in the control of bilinear systems. It is not intended as an exhaustive literature review.

The simplest approach to the control of bilinear systems is to linearise the bilinear equations about the plant's current operating point and then to use the wealth of control theory that exists for linear systems. This approach does not make full use of the non-linear nature of the problem and is an extension of the gain scheduling approach in which a schedule of controller settings is calculated for different plant operating conditions.

Goodwin and Sin (1984) have used a simple dead-beat controller with bounded inputs in simulation studies on the control of acid waste water pH and of oxygen content in a water treatment plant. Svoronos et al (1981) have presented a self tuning regulator for a single-input single-output bilinear system. This used a recursive maximum likelihood estimator with a minimum variance controller, and was applied in a simulation study on the control of a chemical reactor. The methods used by both of the above groups of researchers include feedfoward action to compensate for measured input disturbances, and both groups found that controllers based on bilinear models gave superior results to those based only on linear models.

Mohler (1973) used optimal control theory to deal with several practical examples including a nuclear reactor, rocket trajectories for minimum fuel consumption, ecological, physiological and a socioeconomic problems. He used Pontryagin's maximum principle and dynamic programming to find the control inputs which gave optimum trajectories for bilinear systems.

The popularity of optimal control for linear systems is largely based on the fact that the use of a quadratic performance index and linear feedback of the states results in a Riccati equation which can be solved to provide the coefficients of the feedback vector. Unmodified, such a control strategy is unsuitable for bilinear systems because it does not change the controller action with system operating region, and hence, is not globally stable. Derese and Noldus (1980) overcame this by including the desired stability region as a design specification. Ionescu and Monopoli (1975) demonstrated the global stability for a bilinear system with a control law quadratic in the states.

Derese and Noldus noted the possibility of multiple equilibrium points if a linear in the states control law is used with a bilinear system. Benallou et al (1983) presented a method

for determining the number of these equilibrium points. These difficulties only arise because for a bilinear system with this particular control law the equations to determine the system's steady state values are a set of simultaneous quadratic equations. For an open loop bilinear system, the equivalent set of equations are linear and the problem does not arise. Hence, the matter was not pursued in this work.

CHAPTER 6

A BILINEAR CASE STUDY

6.1 INTRODUCTION

This chapter deals with a case study into the identification of a simple single-input single-output (SISO) bilinear heated tank system. Little work has been done in the past on the identification of simple bilinear systems and even less on multi-input multi-output (MIMO) systems. It was necessary to develop techniques and expertise by tackling a simple system before attempting a complicated system such as a distillation column.

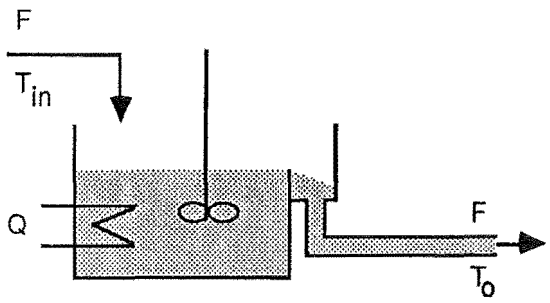
The aim of the case study was to gain experience with bilinear systems, and to develop parameter estimation techniques and means of verifying the resultant models. The estimation techniques had to be robust enough to deal with such practical problems as modeling errors, noise corrupted data and dead time. It was considered advantageous if the techniques were computationally inexpensive and capable of being used in real-time to provide tracking of slowly time varying parameters.

6.2 THE SIMULATED TANK SYSTEM

As mentioned in the previous chapter, the situation often arises in chemical process control where the plant is a fixed volume and the flow through that plant is either a control variable or a measured disturbance. The derivation of the equations describing a mass or energy balance around such a plant demonstrates that these systems fall into the class of systems termed bilinear. This preliminary investigation was limited to a

single compartment with one flowrate through the compartment as the input variable.

The simulated process, shown in Figure 6.1, consisted of a stirred tank of constant volume V, with a constant heat input Q, and a constant inlet temperature T_{in}. The flowrate F through the tank was the input variable and outlet temperature T_o the output variable.



V (volume) = 40 l

Q (heat in) = 14kW

T_{in} = 20°C

F (flowrate) 10 to 15 l/min

T_o = 40°C @ F = 10 l/min

T_o = 33.3°C @ F = 15 l/min

Figure 6.1

The simulated process can be represented graphically as in Figure 6.2 . The first order bilinear block resulted from an energy balance around the tank. The linear first order filter simulated imperfect mixing and temperature sensor response, and was arbitrarily given a time constant of 1 min. The dead time element accounted for the delay in the water reaching the temperature sensor along the outlet pipe, and again was arbitrarily given a value of 0.5 min. The samplers and zero order hold simulate a typical digital data acquisition or control unit.

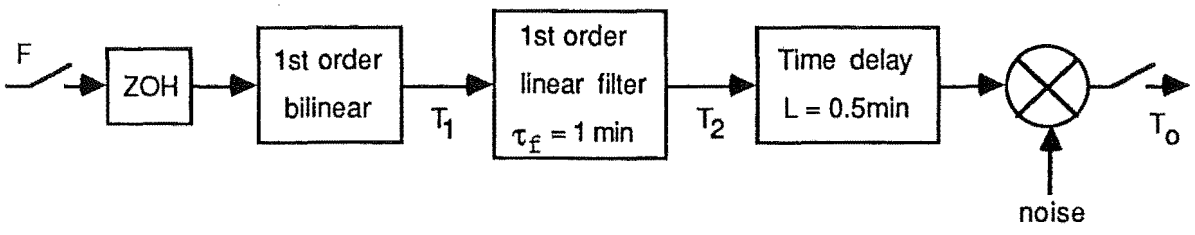


Figure 6.2

For the bilinear element, an energy balance around the tank gives:

$$V\rho C_p \frac{dT_1}{dt} = Q - \rho C_p F(T_1 - T_{in}) \tag{6.1}$$

$$\frac{dT_1}{dt} = \frac{1}{V\rho C_p} Q - \frac{F}{V}(T_1 - T_{in}) \tag{6.2}$$

This is clearly bilinear because the input and state variables, F and T_1 respectively, appear multiplicatively. For this bilinear element, the apparent time constant τ_p and gain K_p are given by the following relations:

$$\tau_p = \frac{V}{F} \tag{6.3}$$

$$K_p = \frac{-Q}{F^2 C_p \rho} \tag{6.4}$$

These result in the following values at two particular operating points :

Time constant:	at F = 10 l/min	τ_p = 4 min
	at F = 15 l/min	τ_p = 2.67 min
Gain:	at F = 10 l/min	K_p = -2°C.min/l
	at F = 15 l/min	K_p = -0.89°C.min/l

As can be seen, a 33% decrease in flowrate causes a 50% increase in time constant and a 125% increase in process gain.

This would certainly be sufficient to cause a conventional process controller, with a 1.7 gain margin, to provide inferior control.

The relations for the first order linear filter and dead time elements can be represented by :

$$\frac{dT_2}{dt} = \frac{1}{\tau_f}(T_1 - T_2) \tag{6.5}$$

$$T_o(t) = T_2(t-L) \tag{6.6}$$

6.3 EXPERIMENTAL

The system described in the previous section was simulated on a VAX minicomputer. A modified Euler method, with a fixed step length of 0.01 minutes, was used to integrate Equations 6.2 and 6.5.

The sampling time was set at 0.5 minutes, which was in the range of one tenth to one sixteenth of the major system time constant. Most of the identification runs were 60 minutes long, which was fifteen to twenty times the major system time constant.

The bilinear model shown in Equation 6.7 was fitted to the simulated plant data using the method suggested in Section 5.4. This method required that the model structure, order, and dead time were specified *a priori*. These factors were varied to show the effect of each on the quality of the overall fit.

$$\begin{aligned} T_o(k) + a_1T_o(k-1) + \ldots + a_nT_o(k-n) \\ = b_1F(k-1-d) + \ldots + b_nF(k-n-d) \\ + c_1{}^1T_o(k-1)F(k-1-d) + \ldots + c_1{}^nT_o(k-1)F(k-n-d) \\ + c_2{}^1T_o(k-2)F(k-1-d) + \ldots + c_2{}^nT_o(k-2)F(k-n-d) \\ + \ldots\ldots \\ + c_n{}^1T_o(k-n)F(k-1-d) + \ldots + c_n{}^nT_o(k-n)F(k-n-d) + DC \end{aligned} \tag{6.7}$$

The least squares algorithm, as shown in Equations 4.12 to 4.14, was initially used to estimate the parameters, with the parameter and state vector defined as :

$$\theta_o^T = [a_1, a_2 \ldots a_n, b_1 \ldots b_n, c_1^1, c_1^2 \ldots c_n^n, DC]$$

$$\varphi(k)^T = [-T_o(k-1), -T_o(k-2) \ldots -T_o(k-n), F(k-1-d) \ldots F(k-n-d), T_o(k-1)F(k-1-d) \ldots T_o(k-n)F(k-n-d), 1]$$

This was adequate for the linear and low order bilinear models. For the higher order bilinear models, it required the use of high precision arithmetic and sometimes still gave unpredictable results or became unstable. Consequently, the U-D algorithm, as shown in Section 4.7, was used for all the results presented in this chapter. It gave adequate results in single precision for situations in which the least squares algorithm did not converge.

The U-D algorithm was a recursive identification algorithm, but in this study, it was used in a batchwise manner. This meant that it could later be used in real-time to follow changes in process parameters.

Three means were employed to verify the identified models. Neither of these means was sufficient by itself but when taken together were adequate. All the elements of the simulated system shown in Figure 6.2, except the first order bilinear element, had a unity gain. Hence, it was possible to compare the process gain of the identified model at different operating points with that of the bilinear element at the same points. The off-line nature of this study meant that it was possible to evaluate the variance of the difference between simulated plant and model outputs. These variances could be directly compared for situations in which no measurement noise was present, but otherwise the measurement noise variance had to be taken into account. Lastly, it was possible to

visually compare the graphs of the plant output with that of the model for the same input function.

6.4 RESULTS AND DISCUSSION

6.4.1 INPUT FUNCTION

The form of the input was important as it needed to excite the non-linearities in order to identify them. A number of different functions were tried, including square waves, and single or multiple steps, all with or without a superimposed random binary sequence. The best function was found to be a multiple step with a superimposed random sequence. The multiple step was necessary for the correct identification of the change of process gain with operating point and the random binary sequence to be able to distinguish between the different b parameters.

Two models of the same form were fitted to two different sets of I/O data to demonstrate the importance of the input function. Graph 6.1 shows an input function consisting of a square wave and Graph 6.3 shows a double step with a random binary sequence superimposed. Graphs 6.2 and 6.8, respectively, show the resultant second order diagonal bilinear fitted models. A comparison of the fitted model gains for these graphs in Table 6.1 shows that the double step with random sequence resulted in a closer fit to the plant gains.

6.4.2 MODEL CHOICE

The Graphs 6.4 to 6.11 consist of models of increasing complexity fitted to the I/O data generated using Graph 6.3 as the plant input with no measurement noise. The parameter estimation algorithm was given the correct amount of dead time *a priori*. The aim of these experiments was to demonstrate the ability of different models to emulate the plant, rather than to test the parameter estimation algorithm.

A comparison of Graphs 6.4 to 6.6 with Graphs 6.7 to 6.11 shows that the performance of the linear models was inferior to that of the bilinear models. This was expected because the linear models were unable to follow the changes of plant parameters with operating point and simply increasing the model order could not compensate for the model deficiencies.

Increasing the model order or the number of terms in the model improved the closeness of the overall fit, as shown in Graphs 6.4 to 6.11. The simplest model which gave results of the accuracy required for control was the second order diagonal bilinear model. The improvement of fit due to using full bilinear models rather than diagonal was small and did not justify the increased model complexity.

For Graphs 6.12 to 6.15, the parameter estimation algorithm was no longer given the correct amount of dead time. The aim was to test the resistance of the combination of algorithm and different model forms to uncertain dead times. The graphs represent 2nd and 3rd order diagonal bilinear models for the situations in which the dead time has been either over or under estimated by 0.5 min. The results were satisfactory and showed that to a certain extent this uncertainty could be compensated for by using a higher order model.

6.4.3 NOISE IMMUNITY

Graphs 6.16 and 6.17 use the same model as in Graph 6.8 but with increasing amounts of measurement noise, with the objective of testing the sensitivity of the parameter estimation algorithm to noise. The addition of noise certainly downgraded the performance of the algorithm. As shown in Table 6.1, measurement noise with a standard deviation of 0.1°C resulted in up to 15% error in the model gains.

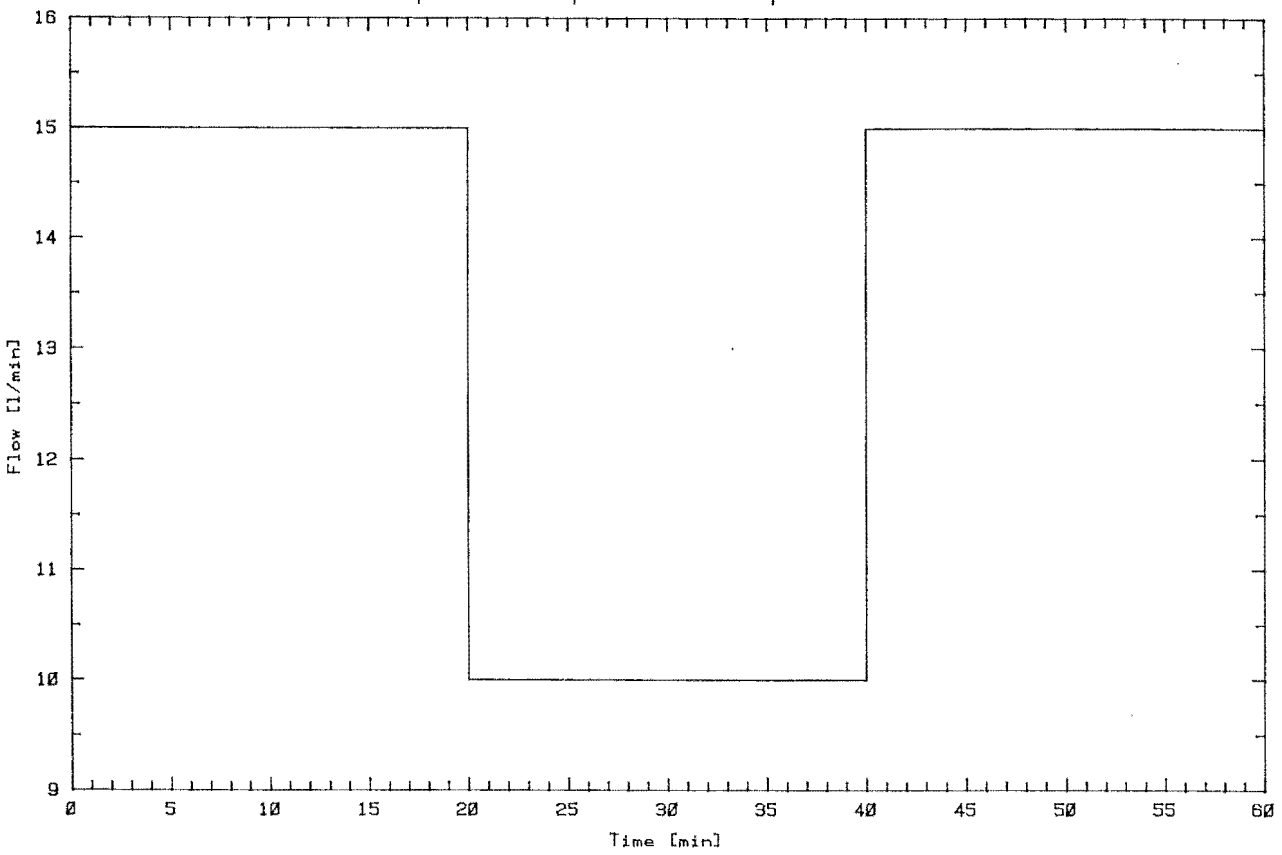
Graph No	Model Description	Variance (1)	Model Gain (2) at Different F		
			10l/min	12.5l/min	15l/min
2	2nd order diagonal bilinear	1.2E-4	-1.83	-1.295	-0.976
4	1st order linear	0.16	-1.36	-1.36	-1.36
5	2nd " "	0.050	-1.31	-1.31	-1.31
6	3rd " "	0.037	-1.24	-1.24	-1.24
7	1st order bilinear	0.041	-2.99	-1.277	-0.641
8	2nd order diagonal bilinear	2.7E-4	-2.03	-1.272	-0.869
9	3rd " " "	2.5E-4	-2.02	-1.272	-0.871
10	2nd order bilinear	1.5E-5	-2.01	-1.279	-0.883
11	3rd " "	4.3E-6	-2.01	-1.279	-0.885
12	As G8 but overestimate dead time by .5min	1.6E-2	-2.26	-1.245	-0.765
13	As G9 " " " " " "	6.7E-3	-2.19	-1.255	-0.796
14	As G8 but underestimate dead time by .5min	2.8E-3	-1.90	-1.262	-0.902
15	As G9 " " " " " "	5.1E-4	-2.03	-1.269	-0.864
16	As G8 but measurement noise SD = 0.01°C	6.4E-4	-2.06	-1.274	-0.860
17	As G8 " " " SD = 0.1°C	1.9E-2	-2.31	-1.271	-0.780
18	As G17 but 3 times as many data points	3.7E-2	-2.04	-1.310	-0.913
19	As G17 " 10 " " " " "	4.1E-2	-2.12	-1.320	-0.897

Notes: 1) Variance of difference between plant and model outputs

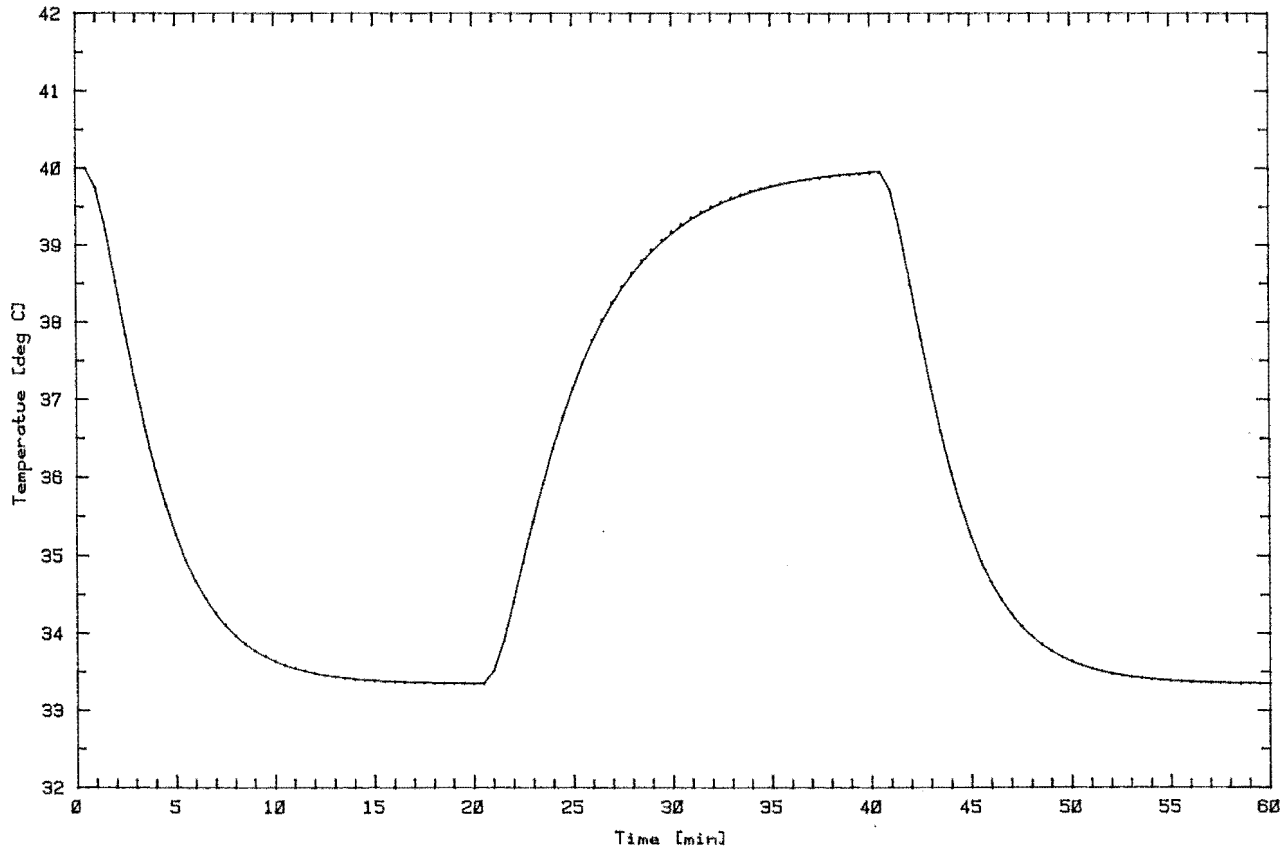
2) Simulated plant gains : @F = 10l/min Kp = -2.00°C.min/l,
@F = 12.5l/min Kp = -1.280°C.min/l,
@F = 15l/min Kp = -0.889°C.min/l

Table 6.1 Model Verification Data

Graph 6.1 Square Wave Input Function

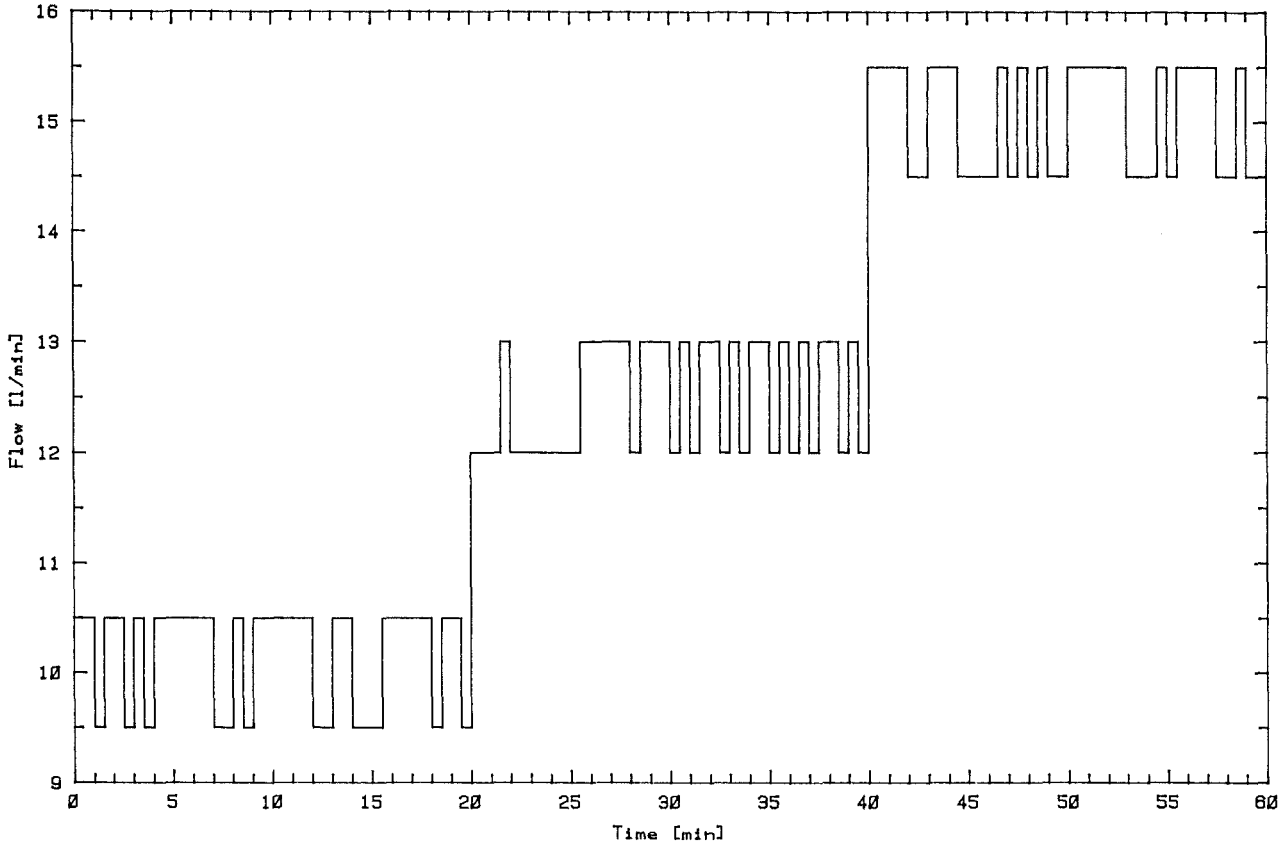


Graph 6.2 2nd Order Diagonal Bilinear Model (Generated using Graph 6.1)

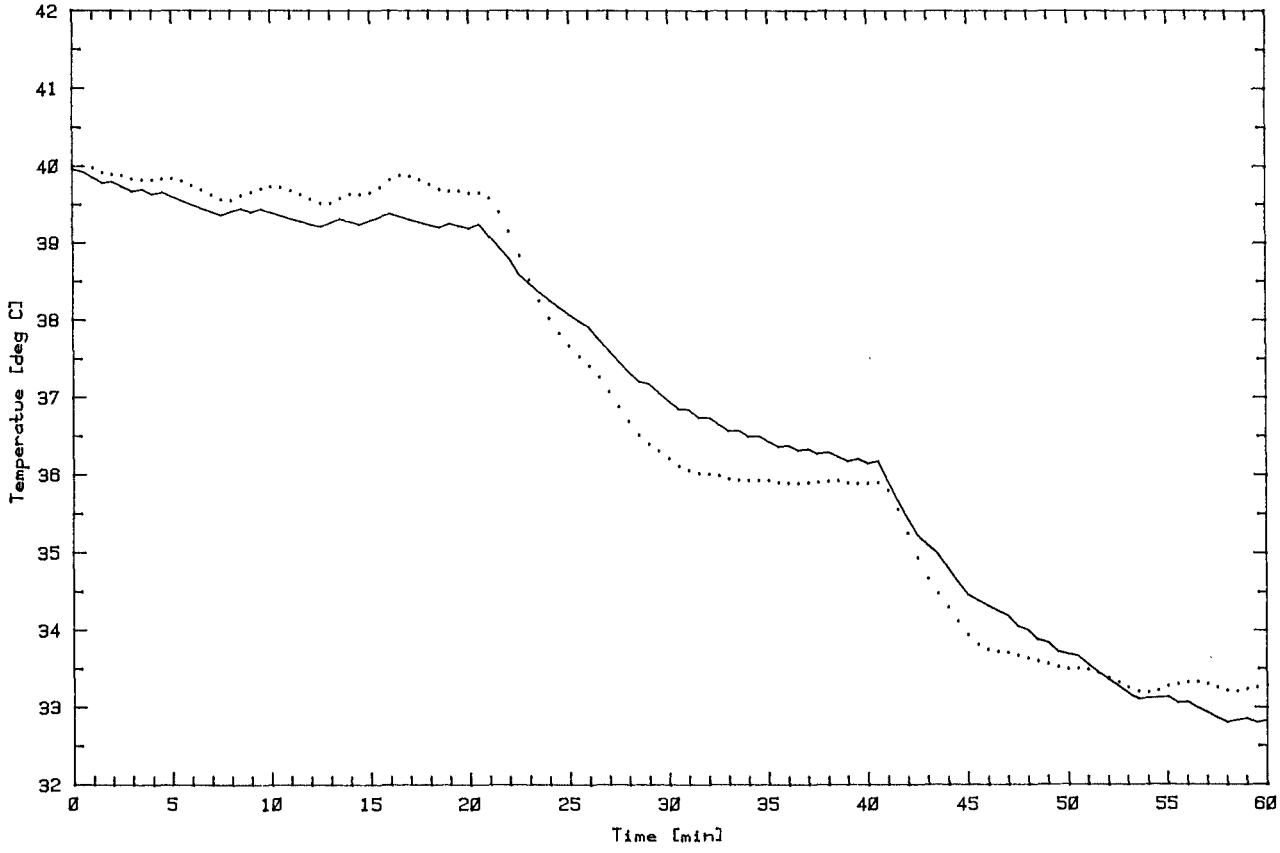


Key: ····· Simulated Plant Output
—— Identified Model Output

Graph 6.3 Input Function (2 Steps & Random Binary Sequence)

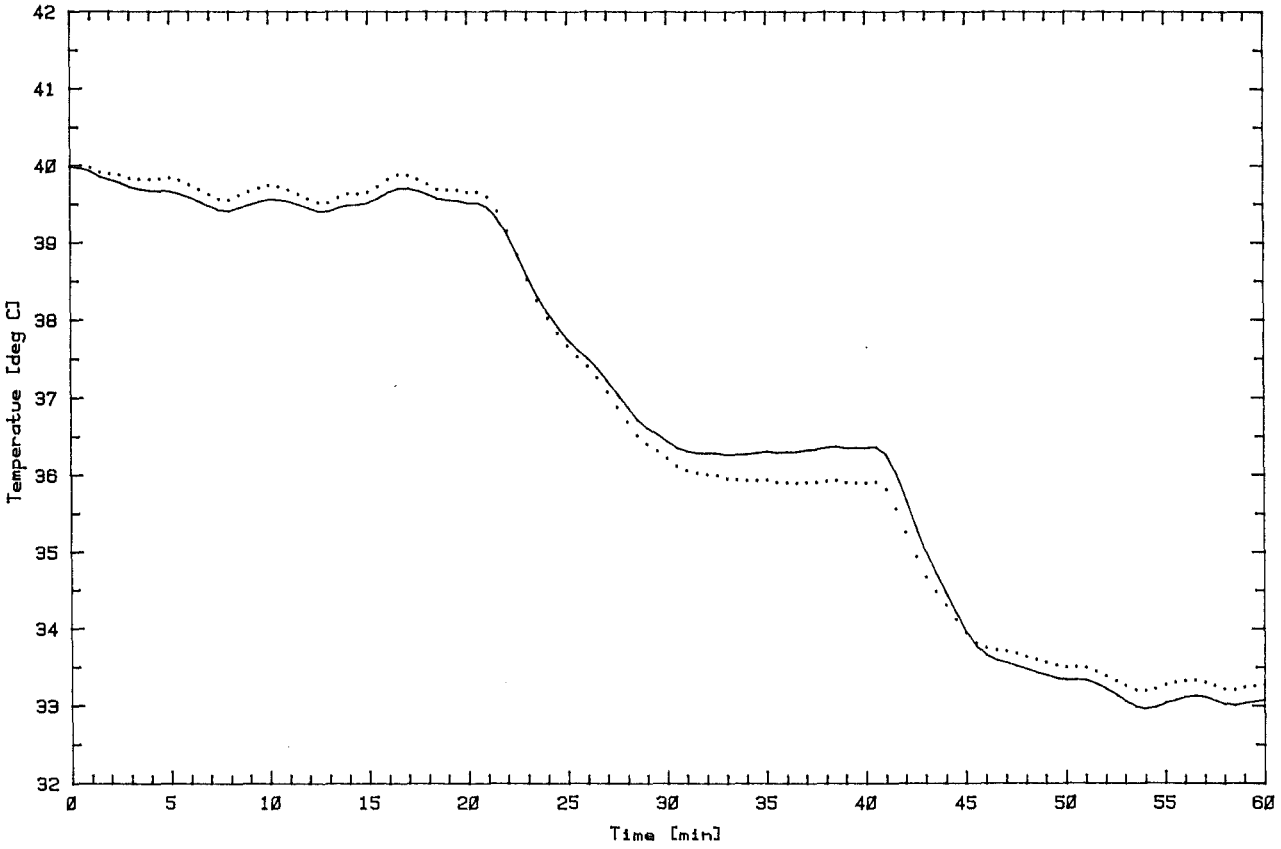


Graph 6.4 1st Order Linear Model

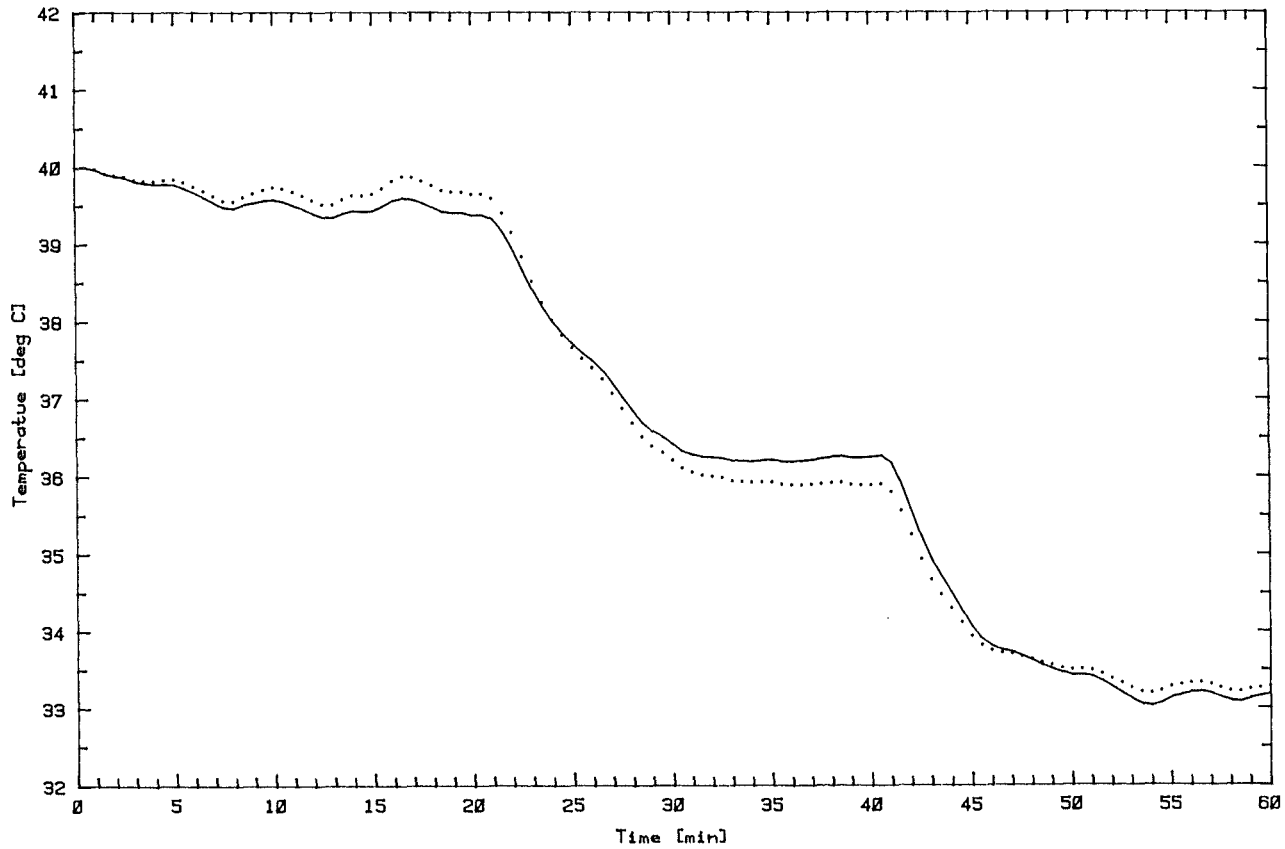


Key: ····· Simulated Plant Output
—— Identified Model Output

Graph 6.5 2nd Order Linear Model

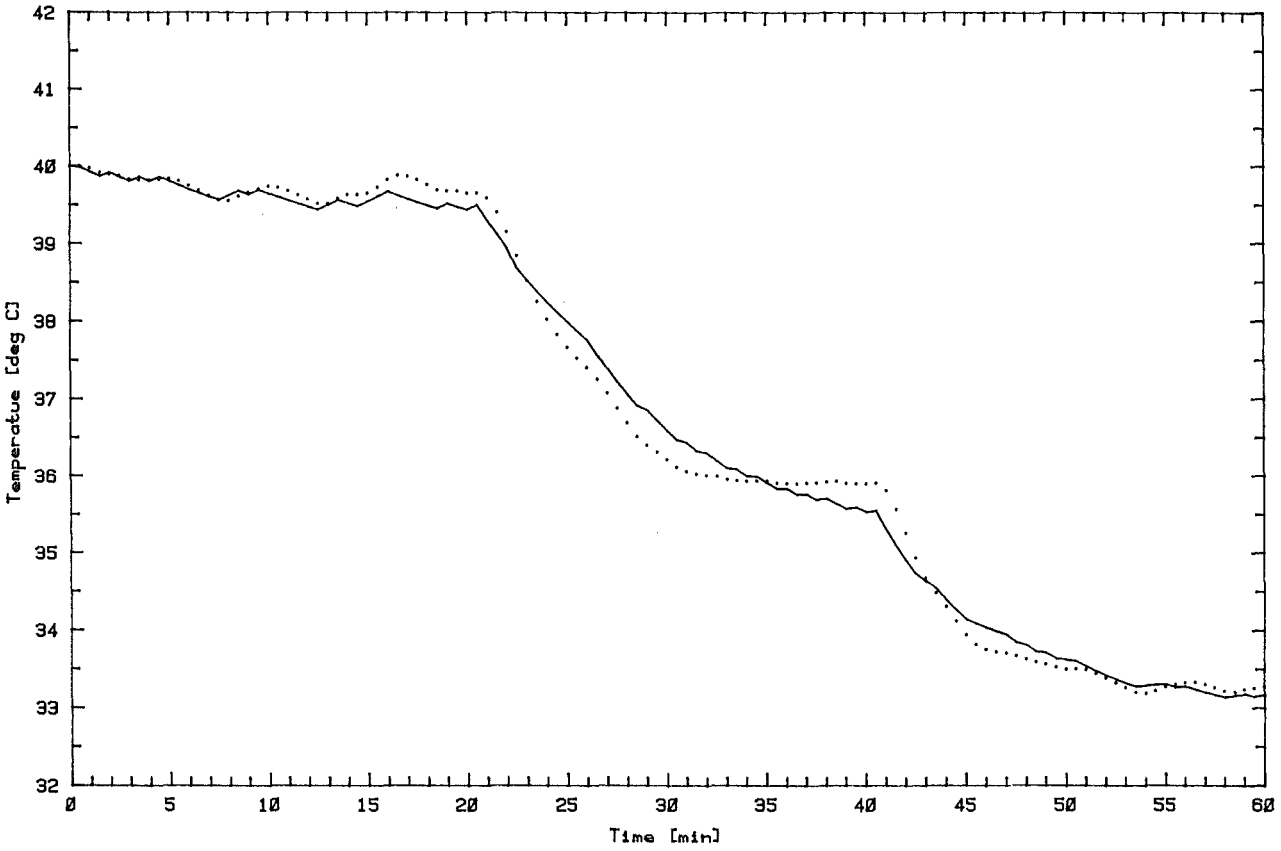


Graph 6.6 3rd Order Linear Model

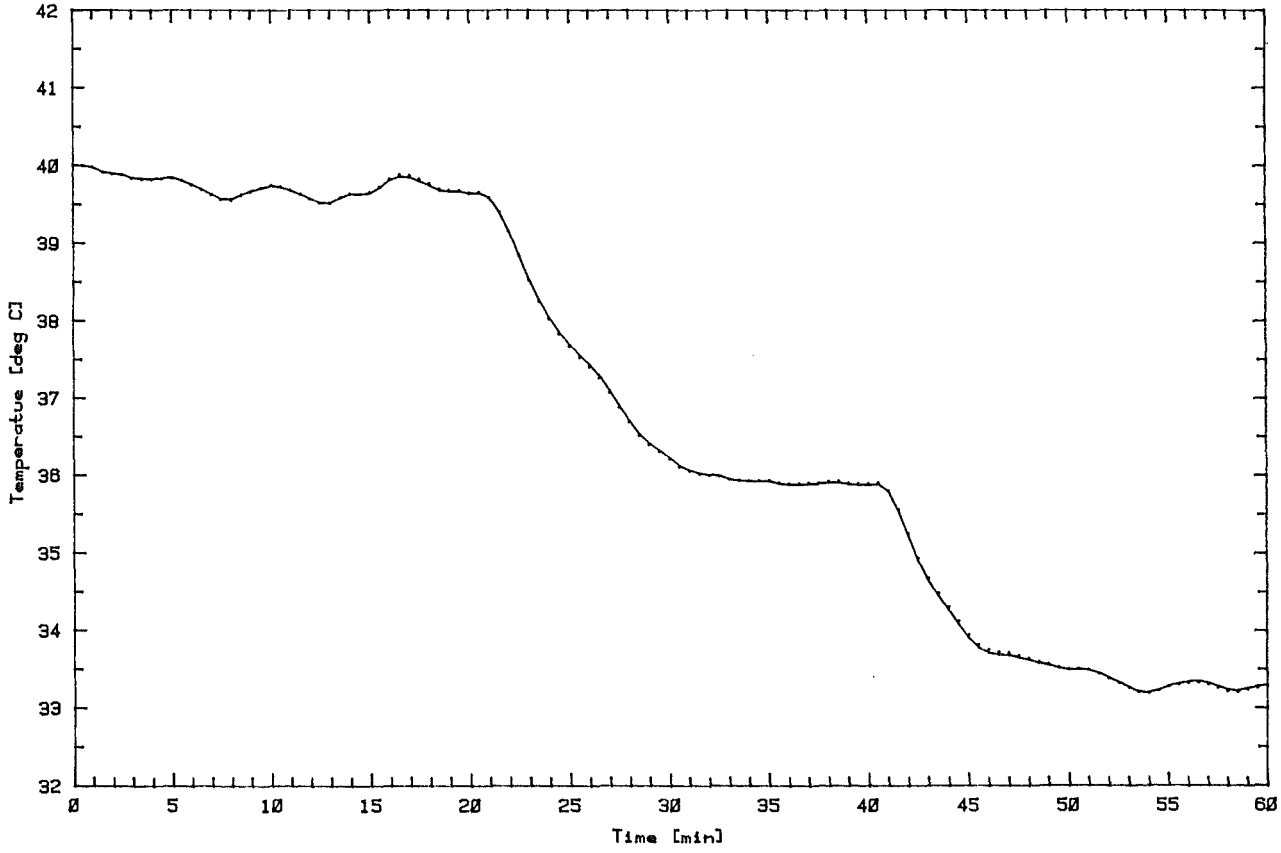


Key: ····· Simulated Plant Output
—— Identified Model Output

Graph 6.7 1st Order Bilinear Model

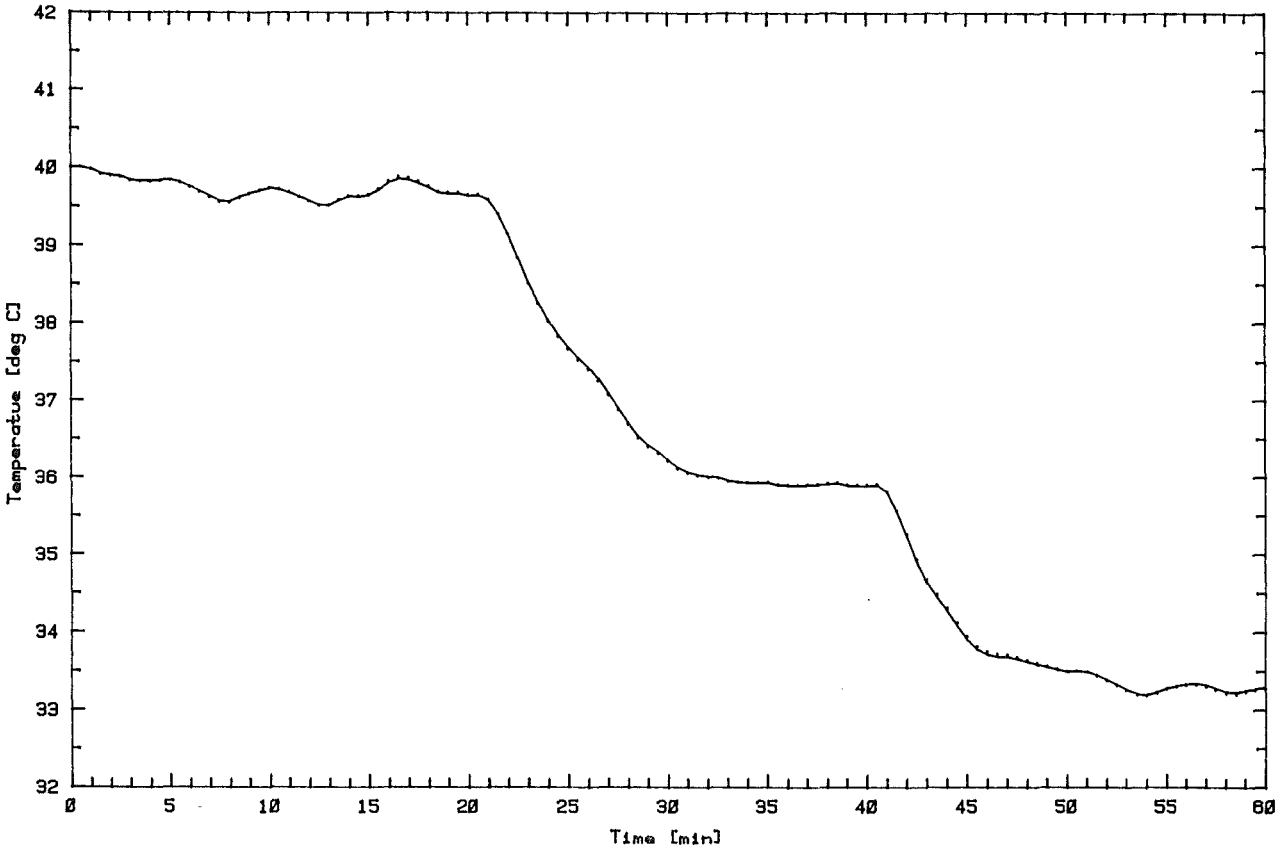


Graph 6.8 2nd Order Diagonal Bilinear Model

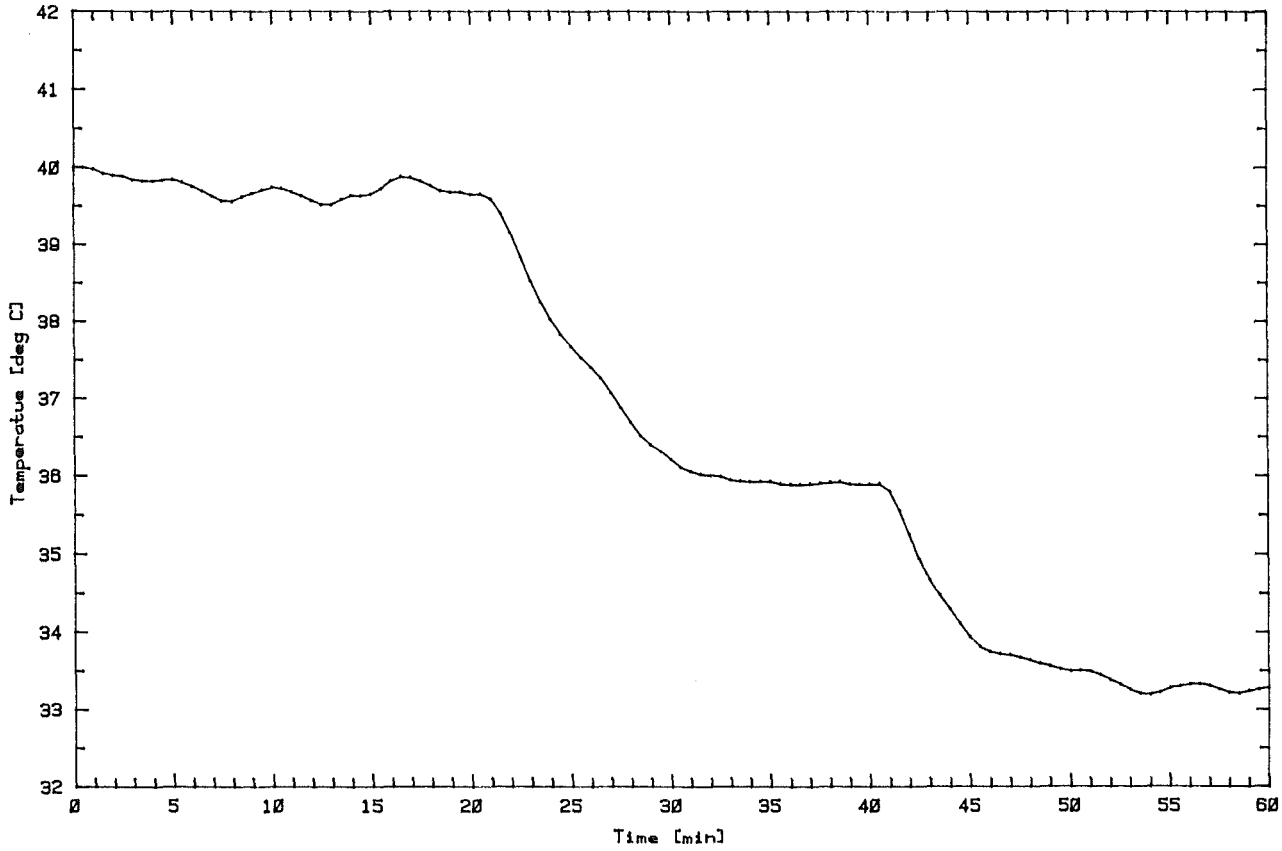


Key: ····· Simulated Plant Output
—— Identified Model Output

Graph 6.9 3rd Order Diagonal Bilinear Model

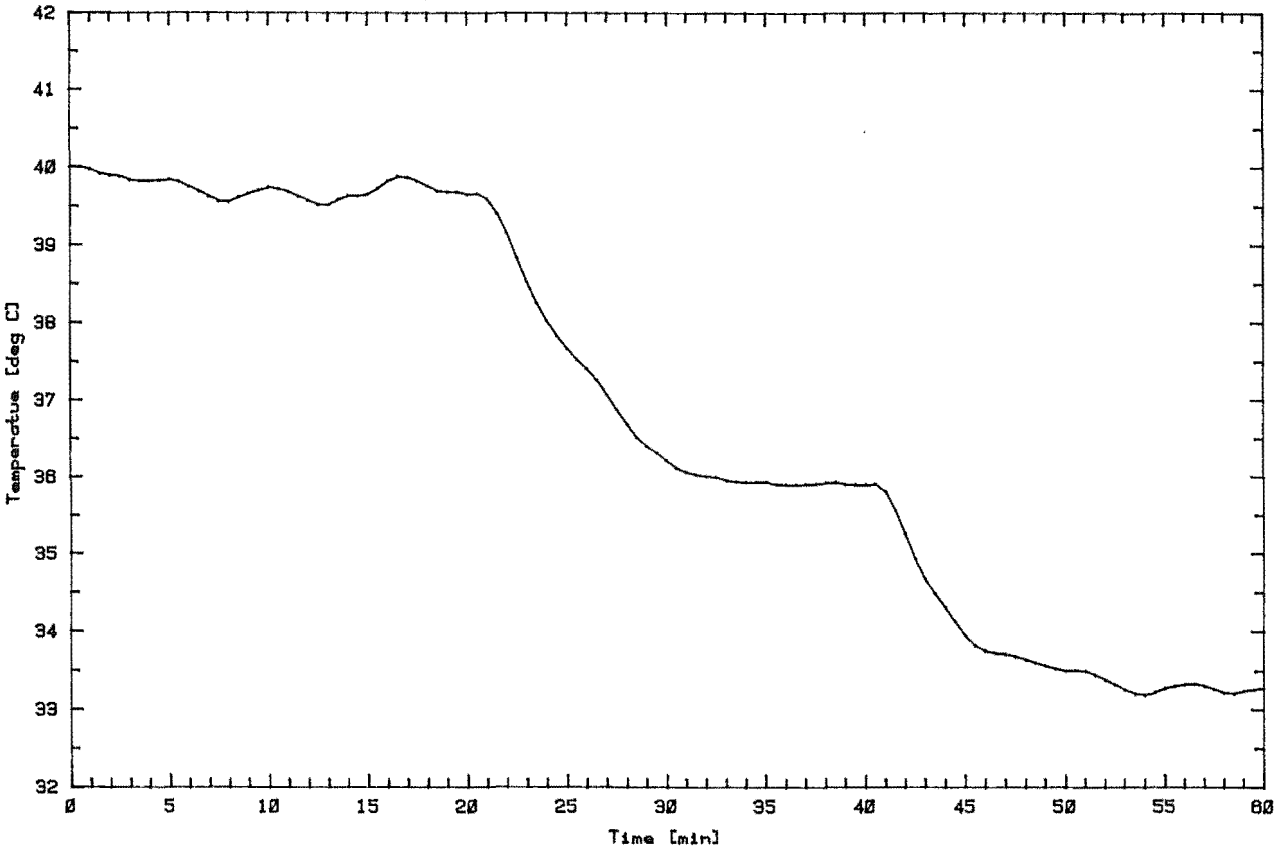


Graph 6.10 2nd Order Bilinear Model

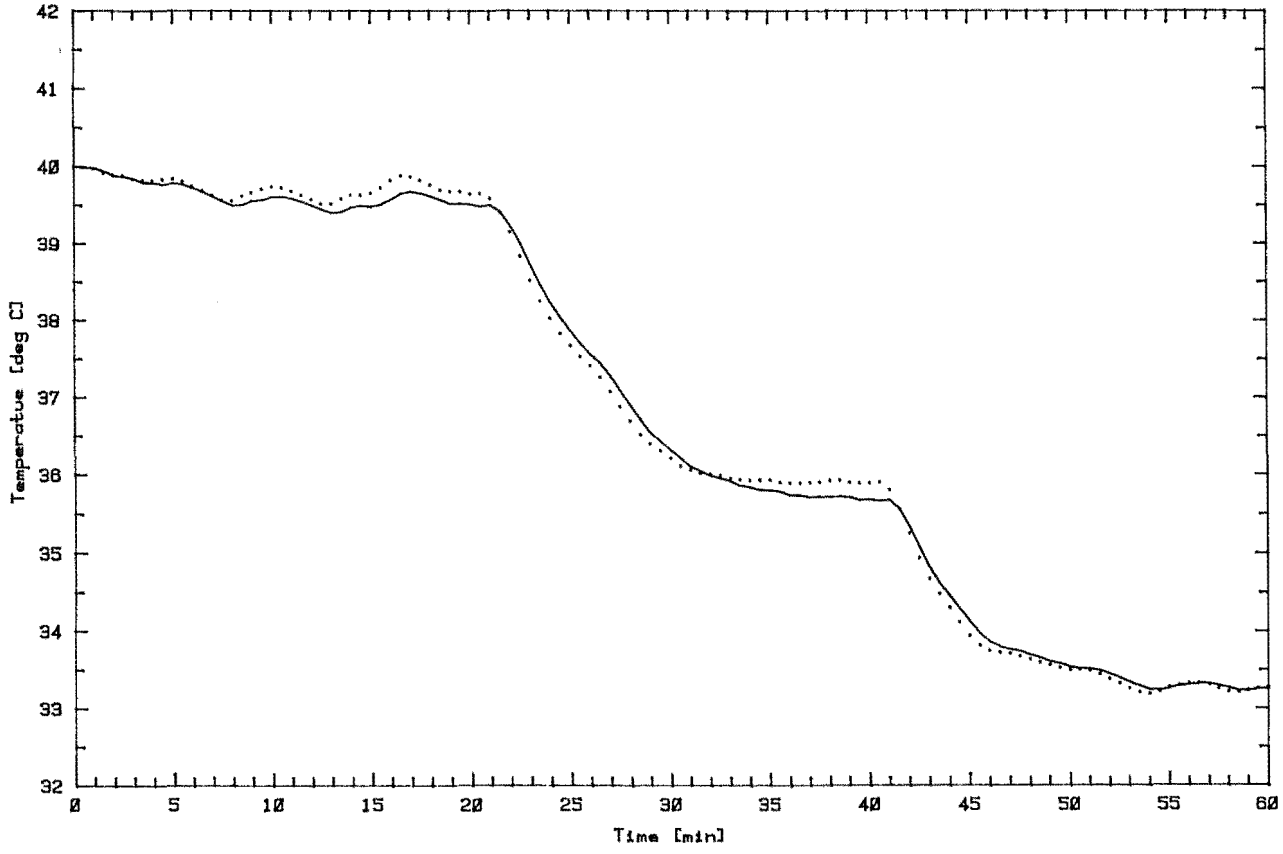


Key: ····· Simulated Plant Output
—— Identified Model Output

Graph 6.11 3rd Order Bilinear Model

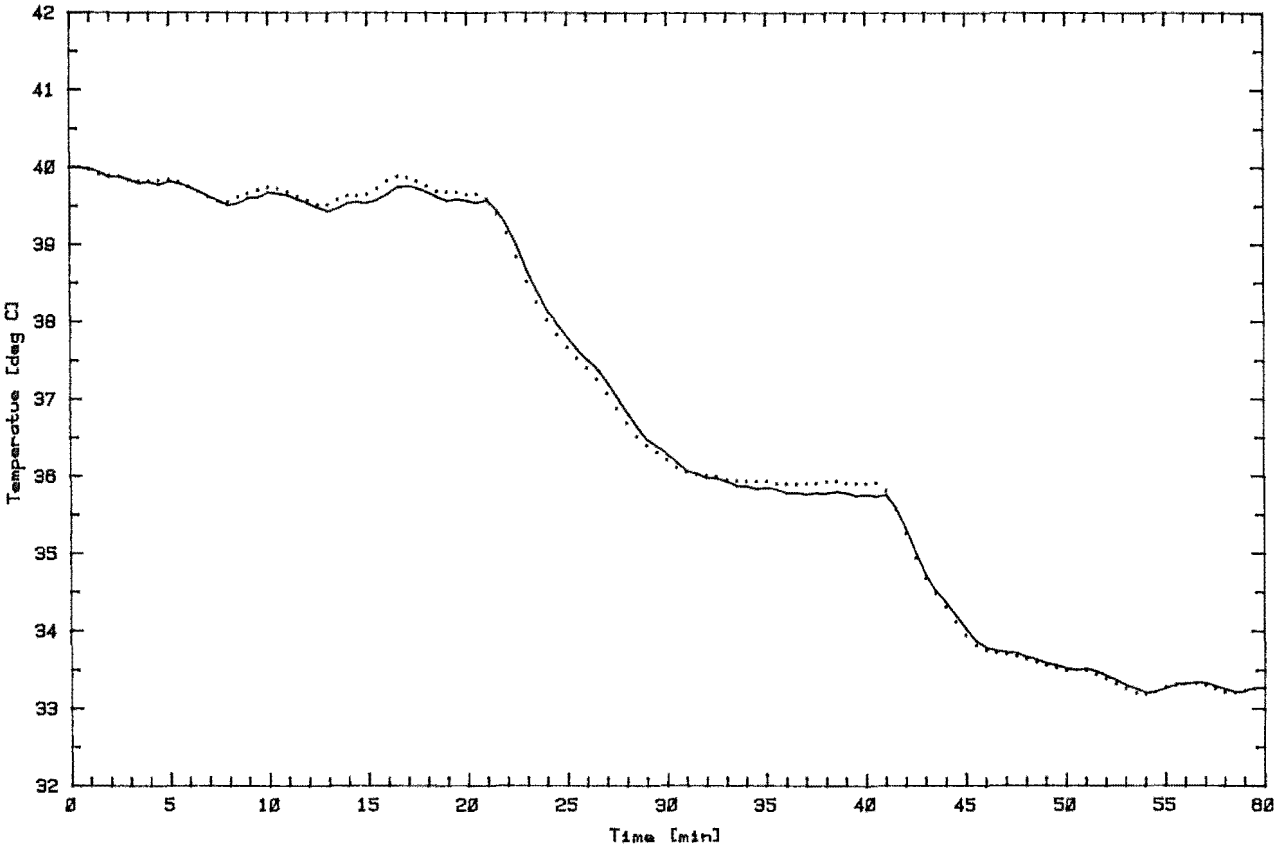


Graph 6.12 2nd Order Diagonal Bilinear Model (over-estimate dead time)

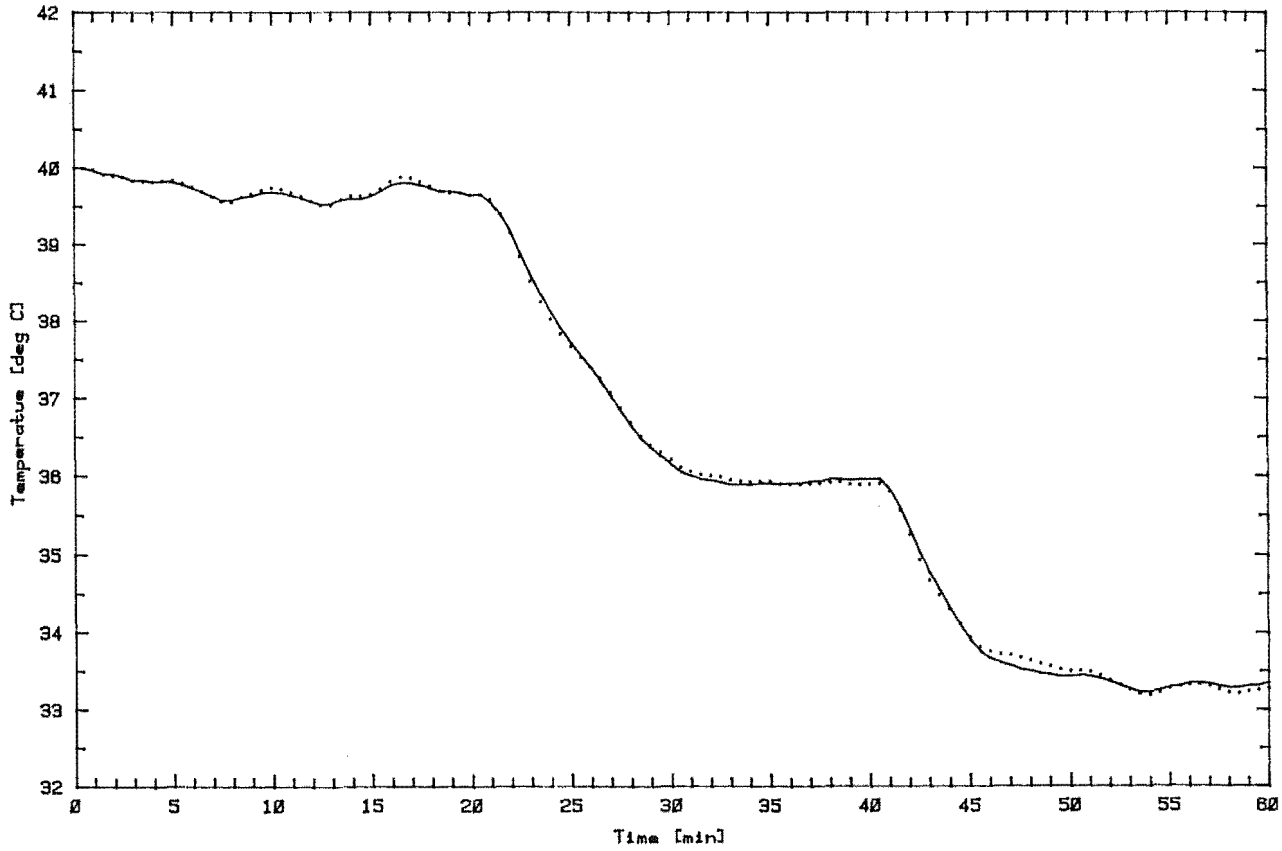


Key: ····· Simulated Plant Output
—— Identified Model Output

Graph 6.13 3rd Order Diagonal Bilinear Model (over-estimate dead time)

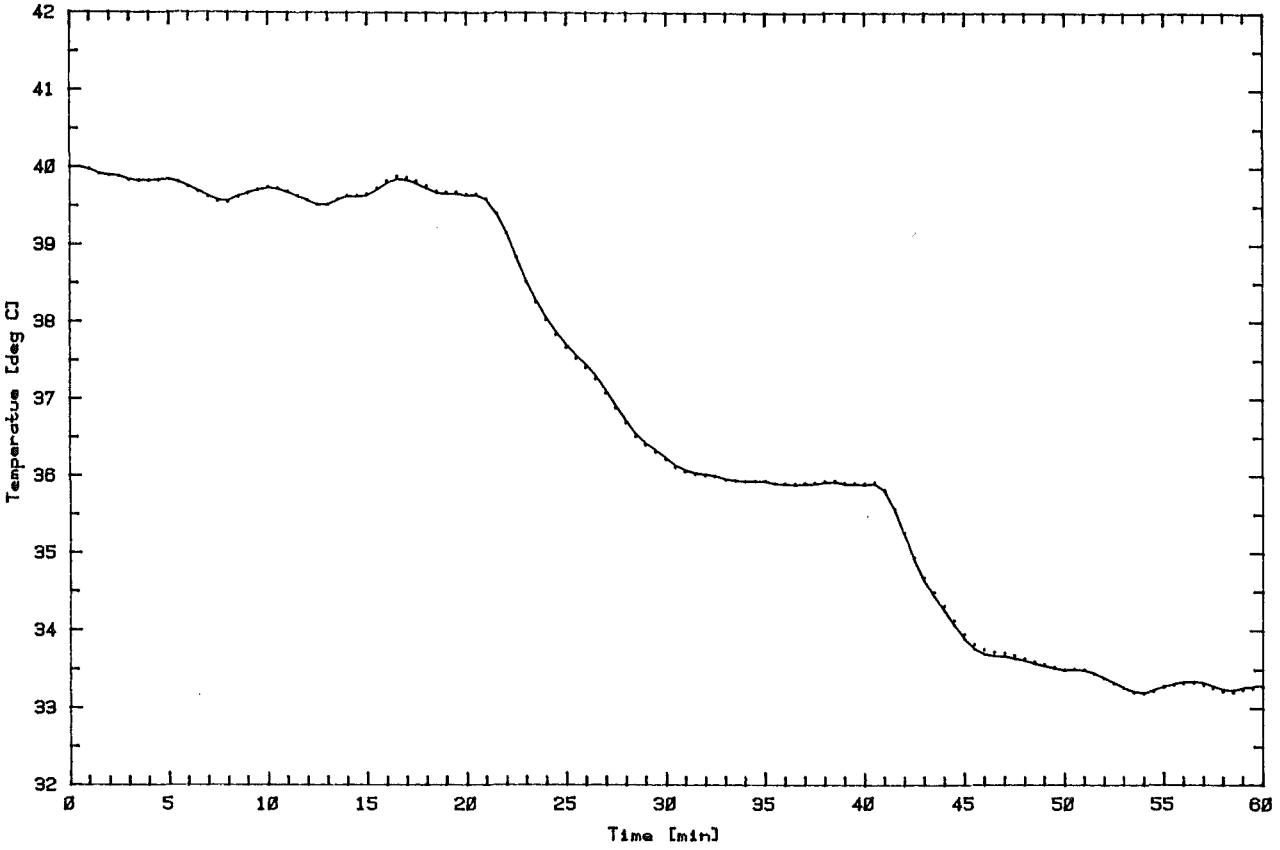


Graph 6.14 2nd Order Diagonal Bilinear Model (under-estimate dead time)

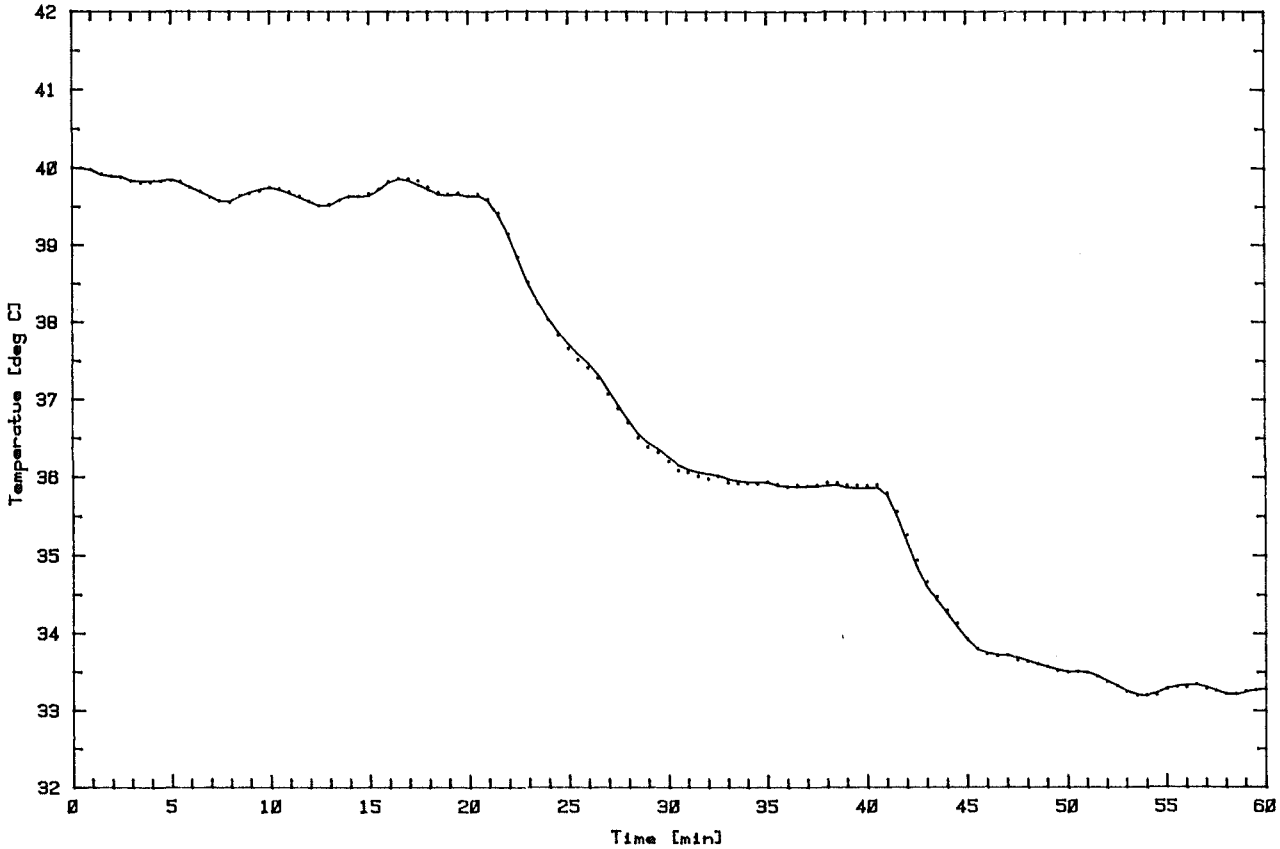


Key: ····· Simulated Plant Output
—— Identified Model Output

Graph 6.15 3rd Order Diagonal Bilinear Model (under-estimate dead time)

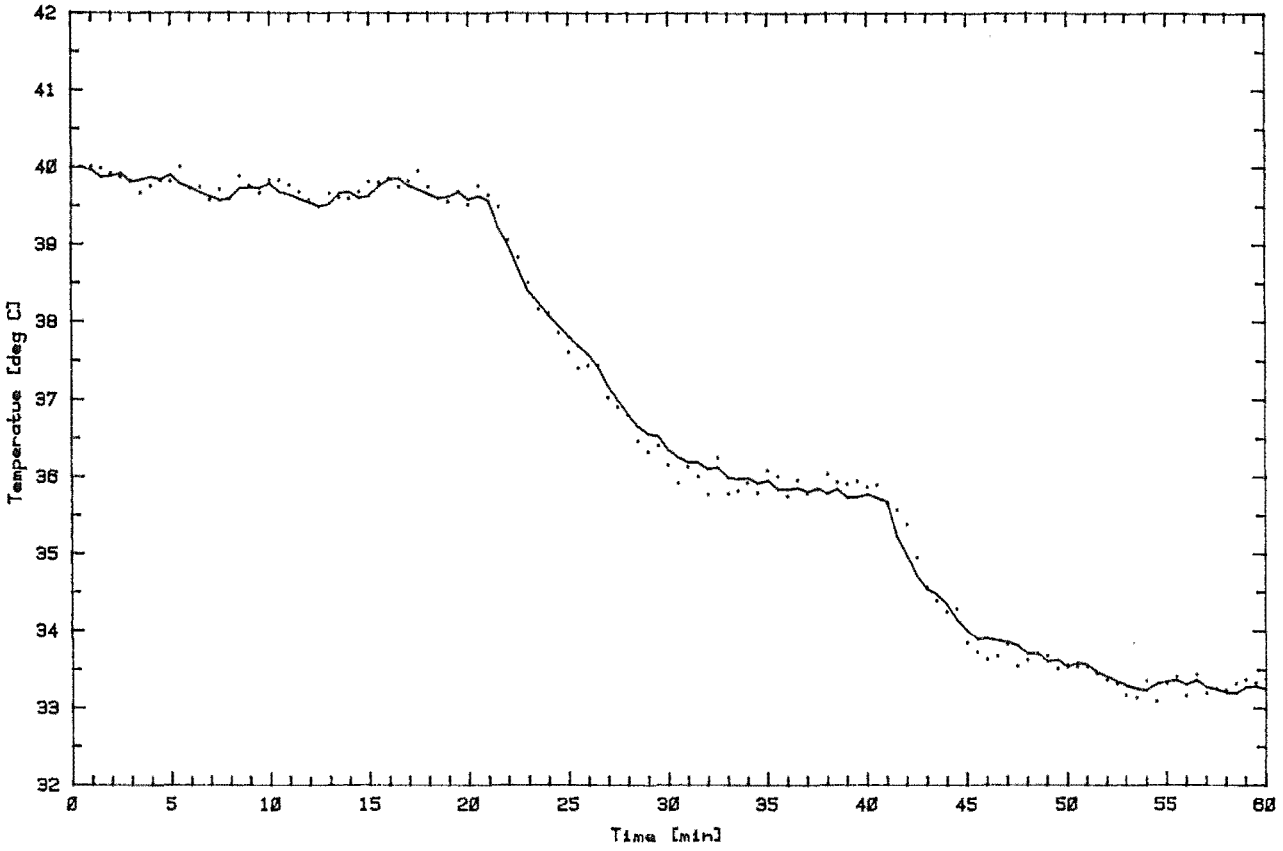


Graph 6.16 2nd Order Diagonal Bilinear Model (noise SD = 0.01deg C)

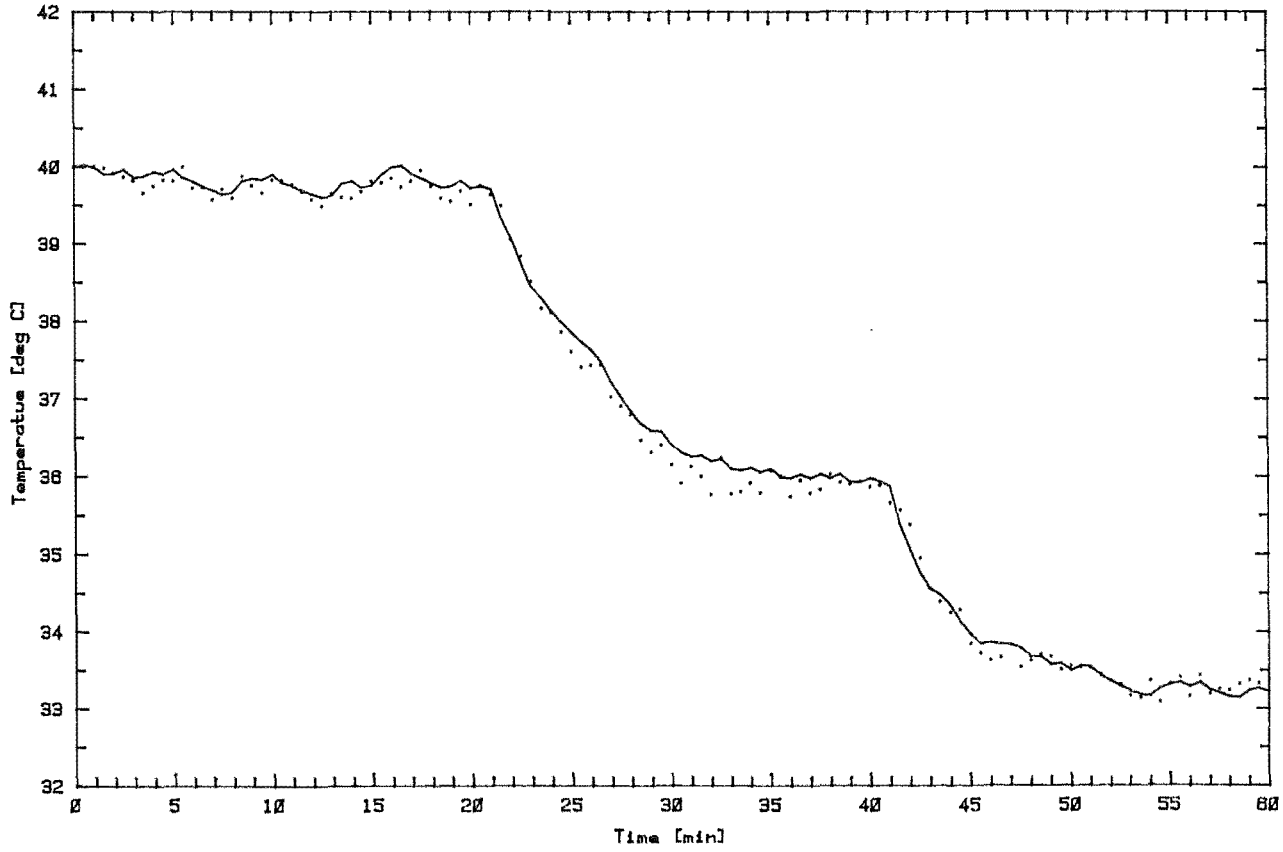


Key: ····· Simulated Plant Output
—— Identified Model Output

Graph 6.17 2nd Order Diagonal Bilinear Model (noise SD = Ø.1deg C)

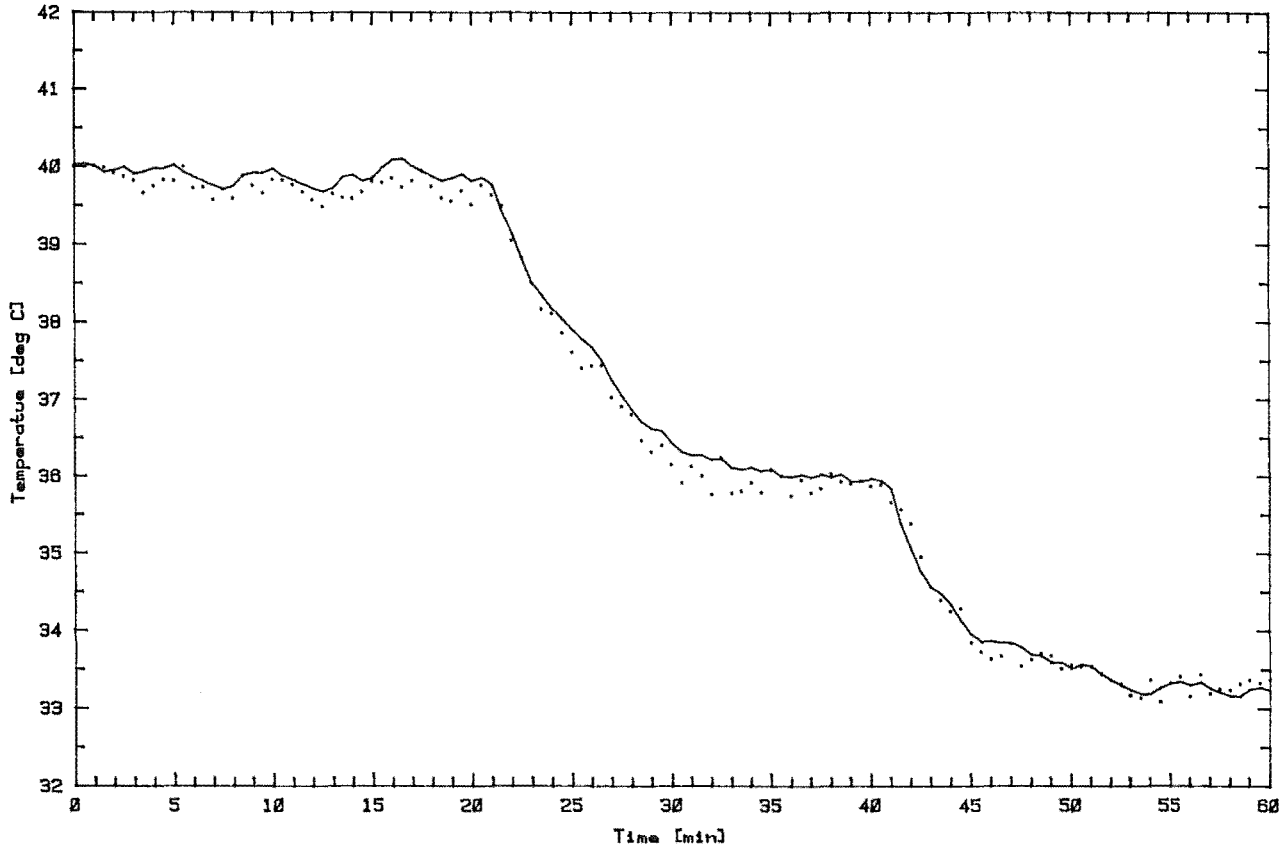


Graph 6.18 Same as Graph 6.17 but 3 times as many data points



Key: ····· Simulated Plant Output
—— Identified Model Output

Graph 6.19 Same as Graph 6.17 but 10 times as many data points



Key: ····· Simulated Plant Output
—— Identified Model Output

Graphs 6.18 and 6.19 were a repeat of Graph 6.17, but with three and ten times, respectively, the number of data points used in the parameter estimation. This was to test if the model degradation with the inclusion of noise was due to a poorer convergence rate, or due to bias in the results. Increasing the number of data points did not help significantly, implying that in the presence of correlated noise, the algorithm converged to a biased solution as expected.

6.5 CONCLUSIONS

Bilinear models were successfully identified for a simulated heated tank system and were shown to be better than linear models for systems of this type.

The U-D algorithm proved to be robust and capable of dealing with modelling errors and noise.

The addition of noise resulted in bias in the identified bilinear model parameters.

The effect of uncertain dead times in bilinear systems could be aliased with high order effects.

This case study showed that there was considerable potential for the application of bilinear models to other chemical processes.

CHAPTER 7

BILINEAR IDENTIFICATION OF A SIMULATED DISTILLATION COLUMN

7.1 BILINEAR MODEL

Distillation columns fall within the class of systems termed bilinear (Espana (1977), Beghelli and Guidorzi (1976)). A distillation column consists of a number of interconnected compartments and the flows through them are related linearly to the input variables. This type of compartmental system is often bilinear.

In this section, a bilinear model of a binary distillation column is derived. The following assumptions are made initially, but as shown later in this section, several of these may be relaxed and have no effect on the essentially bilinear form of the resultant model.

1. Negligible vapour holdup.
2. Liquid well mixed on plates.
3. Equi-molar overflow.
4. Equilibrium of form $y_i = k_i * x_i$ where k_i is a constant for each plate.
5. Feed at its bubble point
6. Plates, reboiler and condenser have constant molar holdups.

Component mass balance for a total condenser :

$$m_0 \frac{dx_0}{dt} = Vk_1x_1 - Vx_0 \qquad (7.1)$$

Component mass balance for the i^{th} plate in the rectification section :

$$m_i \frac{dx_i}{dt} = Vk_{i+1}x_{i+1} + Lx_{i-1} - Vk_ix_i - Lx_i \qquad (7.2)$$

Component mass balance for the feed plate :

$$m_{if} \frac{dx_{if}}{dt} = Vk_{if+1}x_{if+1} + Lx_{if-1} + Fx_f - Vk_{if}x_{if} - Lx_{if} \qquad (7.3)$$

Component mass balance for the i^{th} plate in the stripping section:

$$m_i \frac{dx_i}{dt} = Vk_{i+1}x_{i+1} + (L+F)x_{i-1} - Vk_ix_i - (L+F)x_i \qquad (7.4)$$

Component mass balance for the reboiler :

$$m_{N+1} \frac{dx_{N+1}}{dt} = (L+F)x_N - Vk_{N+1}x_{N+1} - (L+F-V)x_{N+1} \qquad (7.5)$$

When these equations are combined they result in a bilinear state space model of the form :

$$\frac{d\mathbf{x}}{dt} = B_1V\mathbf{x} + B_2L\mathbf{x} + B_3F\mathbf{x} + \mathbf{c}Fx_f \qquad (7.6)$$

where $\mathbf{x}$ and $\mathbf{u}$ are the state and input vectors respectively and are defined as :

$$\mathbf{x}^T = [x_0, x_1, \ldots x_N, x_{N+1}]$$

$$\mathbf{u}^T = [V, L, F, Fx_f]$$

where

$$c_i = 0 \quad \text{for} \quad i \neq if$$

and $c_{if} = 1$

$$B_1 = \begin{bmatrix} \frac{-1}{m_0} & \frac{k_1}{m_0} & & & & & \\ & \frac{-k_i}{m_i} & \frac{k_{i+1}}{m_i} & & & & \\ & & \frac{-k_{if}}{m_{if}} & \frac{k_{if+1}}{m_{if}} & & & \\ & & & \frac{-k_i}{m_i} & \frac{k_{i+1}}{m_i} & & \\ & & & & & & \frac{1-k_{N+1}}{m_{N+1}} \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 0 & & & & & & \\ \frac{1}{m_i} & \frac{-1}{m_i} & & & & & \\ & & \frac{1}{m_{if}} & \frac{-1}{m_{if}} & & & \\ & & & & \frac{1}{m_i} & \frac{-1}{m_i} & \\ & & & & & \frac{1}{m_{N+1}} & \frac{-1}{m_{N+1}} \end{bmatrix}$$

$$B_3 = \begin{bmatrix} 0 & & & & & & \\ 0 & 0 & & & & & \\ & & 0 & 0 & & & \\ & & & & \frac{1}{m_i} & \frac{-1}{m_i} & \\ & & & & & \frac{1}{m_{N+1}} & \frac{-1}{m_{N+1}} \end{bmatrix}$$

Some of the assumptions above can be relaxed without affecting the bilinear nature of the resultant system. These were discussed in detail by Espana (1977) and only a summary is given here. The equilibrium expression for each plate can be replaced by one of the form :

$$y_i = k_i x_i + j_i \qquad (7.7)$$

The equi-molar overflow assumption can be replaced by constant liquid and vapour enthalpies, and constant thermal losses on each plate.

Espana (1977) claimed that the region of validity for bilinear models of distillation column was a truncated cone in the three dimensional space, as shown in Figure 7.1, containing all possible combinations of input functions. This region was determined by considering the limitations of the assumptions made in the previous derivation. The deficiency of the equilibrium relation resulted in the conical shape. The reasoning behind this was that a line through the origin corresponded to a constant composition profile down the column, and that the equilibrium relation was invalid if there was a large deviation from this

Expected Bilinear Model Validity Region

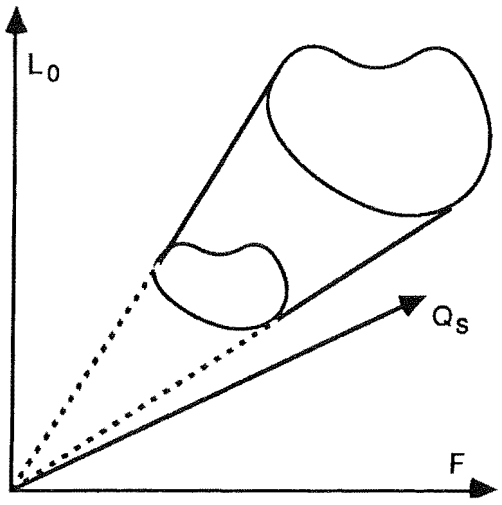


Figure 7.1

profile. The truncation was due to physical constraints such as flooding or weeping and the variation of physical parameters with flowrate.

7.2 IDENTIFICATION

The continuous state space bilinear equation for a distillation column, derived in the previous section, can be converted to a discrete difference equation using the procedure given in Chapter 5. This results in the following equations :

$$x_D(k) = \sum_{i=1}^{n} a_i x_D(k-i) + \sum_{i=1}^{n} b_i Q_S(k-i) + \sum_{i=1}^{n} c_i L_R(k-i)$$
$$+ \sum_{i=1}^{n} d_i F(k-i) + \sum_{i=1}^{n} e_i x_f(k-i) + \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij} x_D(k-i) Q_S(k-j)$$
$$+ \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij} x_D(k-i) L_R(k-j) + \sum_{i=1}^{n}\sum_{j=1}^{n} l_{ij} x_D(k-i) F(k-j) + DC \qquad (7.8)$$

$$x_W(k) = \sum_{i=1}^{n} \underline{a}_i x_W(k-i) + \sum_{i=1}^{n} \underline{b}_i Q_S(k-i) + \sum_{i=1}^{n} \underline{c}_i L_R(k-i)$$
$$+ \sum_{i=1}^{n} \underline{d}_i F(k-i) + \sum_{i=1}^{n} \underline{e}_i x_f(k-i) + \sum_{i=1}^{n}\sum_{j=1}^{n} \underline{g}_{ij} x_W(k-i) Q_S(k-j)$$
$$+ \sum_{i=1}^{n}\sum_{j=1}^{n} \underline{h}_{ij} x_W(k-i) L_R(k-j) + \sum_{i=1}^{n}\sum_{j=1}^{n} \underline{l}_{ij} x_W(k-i) F(k-j) + \underline{DC} \qquad (7.9)$$

where a_i's, b_i's ... l_{ij}'s and $\underline{a}_i$'s, $\underline{b}_i$'s ... $\underline{l}_{ij}$'s are coefficients, and the remainder are discrete time versions of the column variables.

The above equations do not contain any dead time because of the form of the chosen mass and energy balance equations. In practice, dead time may be introduced by sensors such as gas chromatographs or flows in the product lines.

The U-D algorithm was successfully used in the previous chapter to estimate the parameters in a single-input single-output system. For the system of two multi-input single-output equations shown in Equations 7.8 and 7.9, the algorithm was applied separately to the identification of the parameters of each of the two equations.

7.3 RESULTS

The rigorous distillation column simulation described in Chapter 3 was used to generate the input-output data used in this section. This separated the question of the suitability of bilinear models from the problems associated with using experimental data.

The sampling time was set at 4 minutes, because this was the value chosen for the experimental column.

A multiple step function was used for each of the three input functions manipulated; feed, reflux, and steam mass flowrates. The transitions of the input functions were staggered to enable the parameter estimation to distinguish between the effects of the different changes. This resulted in the seven main operating points shown in Table 7.1. A random binary signal was superimposed on the top of the step function in a similar manner to that used in the previous chapter to give the input functions shown in Graph 7.1. The feed composition and thermal condition were kept constant because these variables could not be changed in a controlled manner on the experimental column and it was desirable to be able to compare the results of this simulated study with those of the experimental study in the next chapter.

Column Operating Points

Operating Point No	Column Inputs		
	Feed [l/min]	Steam [kg/min]	Reflux [l/min]
1	1.75	1.10	0.95
2	1.75	1.10	0.75
3	1.75	0.90	0.75
4	1.35	0.90	0.75
5	1.35	0.90	0.65
6	1.35	0.75	0.65
7	1.00	0.75	0.65

Table 7.1

The gains for the simulated column at each of the seven operating points were calculated using finite differences and are shown in Table 7.2. This table shows that as the column flows decrease, the gains tend to increase, which is typical for bilinear systems. Also, the gains of tops composition in response to feed flowrate fluctuate wildly with operating point.

Gains for Simulated Column

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	.022	.127	-.158	.438	.445	-1.16
2	.120	.232	-.382	.412	.366	-0.96
3	.009	.131	-.142	.286	.356	-0.93
4	.059	.199	-.311	.588	.558	-1.43
5	.270	.412	-.779	.350	.283	-0.79
6	.010	.140	-.160	.393	.498	-1.27
7	.214	.358	-.828	.655	.618	-1.30

Table 7.2

Models of several different types and orders were identified using the input/output data generated by the distillation column simulation using the input functions shown in Graph 7.1. The identified model outputs corresponding to the same input functions were then compared with the simulated column outputs in Graphs 7.2 to 7.10. The variances of the differences between the identified model outputs and the simulated column outputs are shown in Table 7.3 . The gains of the identified models at the seven operating points given in Table 7.1 were calculated and are shown in Tables 7.4 to 7.10.

Graphs 7.2 to 7.10 show that bilinear models fit the simulated data better than linear models, but full bilinear models show little improvement over diagonal bilinear models. Table 7.3 supports these observations, and also shows a clear improvement of fit as the order of the bilinear models is increased. A comparison of Tables 7.7 to 7.10 with Table 7.2 shows that the bilinear model gains follow the trend of the simulated column gains, but are unable to follow some of the larger fluctuations.

Variances of Model Errors
for Input Function in Graph 7.1

Model Description	Tops	Bottoms
2nd order linear	$9.4*10^{-6}$	$9.1*10^{-5}$
3rd order linear	$9.0*10^{-6}$	$9.0*10^{-5}$
4th order linear	$8.0*10^{-6}$	$9.0*10^{-5}$
2nd order diagonal bilinear	$4.3*10^{-6}$	$3.5*10^{-6}$
3rd order diagonal bilinear	$3.9*10^{-6}$	$2.3*10^{-6}$
4th order diagonal bilinear	$3.5*10^{-6}$	$1.7*10^{-6}$
2nd order bilinear	$4.4*10^{-6}$	$4.3*10^{-6}$
3rd order bilinear	$3.9*10^{-6}$	$2.5*10^{-6}$
4th order bilinear	$3.6*10^{-6}$	$1.8*10^{-6}$

Table 7.3

Gains of Linear Models

Model	Tops Response			Bottoms Response		
Order	Feed	Reflux	Steam	Feed	Reflux	Steam
2nd	0.028	0.194	-0.215	0.500	0.457	-1.26
3rd	0.029	0.191	-0.216	0.499	0.451	-1.26
4th	0.031	0.189	-0.218	0.500	0.436	-1.24

Table 7.4

Gains of 2nd Order Diagonal Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.019	0.169	-0.186	0.394	0.421	-1.06
2	0.053	0.174	-0.212	0.409	0.391	-0.98
3	0.015	0.151	-0.164	0.368	0.482	-1.22
4	0.034	0.230	-0.258	0.524	0.531	-1.33
5	0.065	0.235	-0.282	0.536	0.505	-1.26
6	0.014	0.203	-0.218	0.485	0.620	-1.57
7	0.039	0.341	-0.376	0.728	0.690	-1.73

Table 7.5

Gains of 3rd Order Diagonal Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.022	0.173	-0.195	0.396	0.423	-1.07
2	0.063	0.168	-0.229	0.416	0.394	-0.99
3	0.014	0.140	-0.154	0.357	0.472	-1.19
4	0.040	0.235	-0.273	0.531	0.535	-1.35
5	0.078	0.229	-0.305	0.547	0.511	-1.29
6	0.013	0.191	-0.203	0.473	0.609	-1.53
7	0.047	0.367	-0.412	0.747	0.701	-1.77

Table 7.6

Gains of 4th Order Diagonal Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.025	0.186	-0.212	0.400	0.424	-1.07
2	0.076	0.157	-0.249	0.423	0.396	-1.01
3	0.013	0.131	-0.144	0.350	0.464	-1.16
4	0.048	0.242	-0.296	0.537	0.536	-1.36
5	0.093	0.217	-0.328	0.556	0.512	-1.31
6	0.011	0.180	-0.189	0.464	0.599	-1.50
7	0.058	0.399	-0.461	0.761	0.704	-1.80

Table 7.7

Gains of 2nd Order Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.019	0.169	-0.186	0.393	0.419	-1.05
2	0.055	0.173	-0.215	0.409	0.391	-0.98
3	0.014	0.148	-0.161	0.368	0.477	-1.22
4	0.034	0.230	-0.258	0.523	0.529	-1.33
5	0.066	0.233	-0.285	0.536	0.506	-1.26
6	0.013	0.199	-0.212	0.486	0.615	-1.56
7	0.040	0.342	-0.377	0.727	0.689	-1.72

Table 7.8

Gains of 3rd Order Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.024	0.185	-0.210	0.396	0.421	-1.06
2	0.072	0.159	-0.243	0.418	0.396	-1.00
3	0.014	0.135	-0.148	0.356	0.467	-1.18
4	0.044	0.238	-0.286	0.531	0.535	-1.35
5	0.085	0.215	-0.314	0.549	0.514	-1.29
6	0.012	0.182	-0.191	0.472	0.603	-1.53
7	0.051	0.373	-0.426	0.749	0.703	-1.77

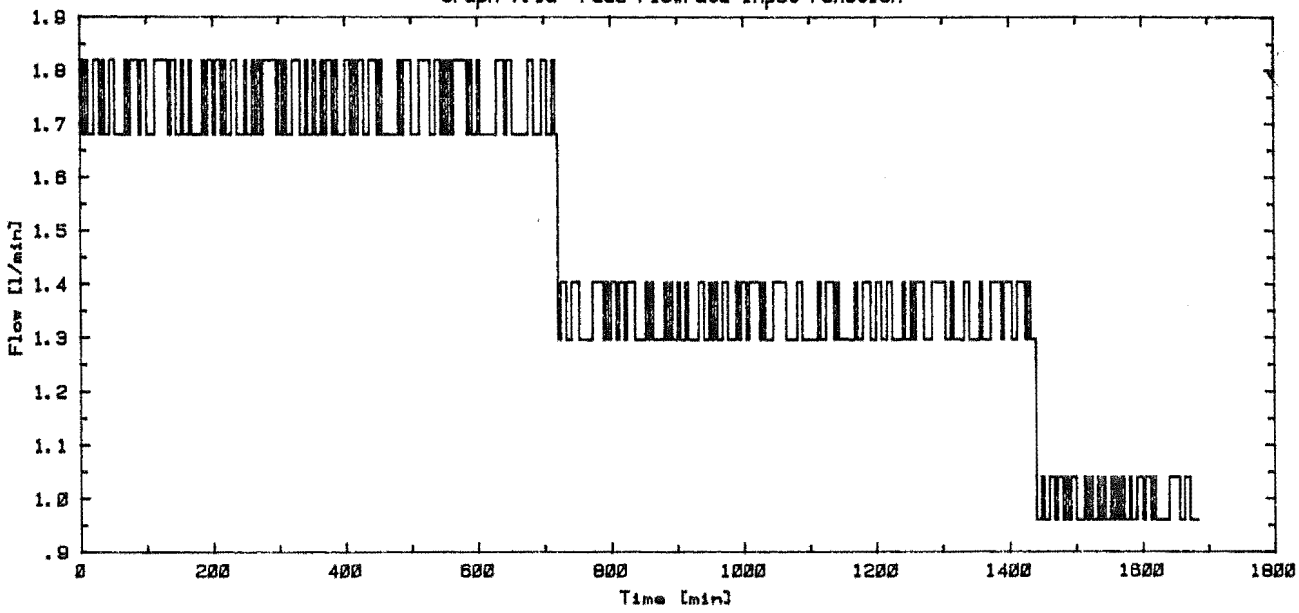
Table 7.9

Gains of 4th Order Bilinear Model

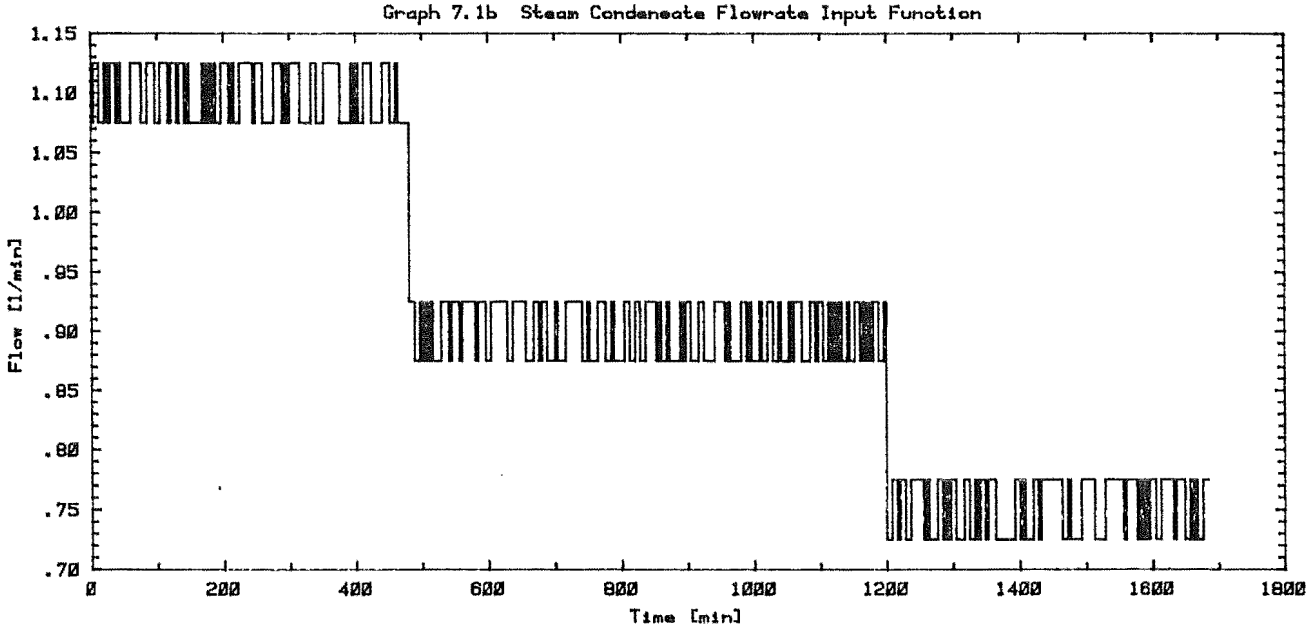
Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.028	0.193	-0.224	0.401	0.425	-1.08
2	0.088	0.153	-0.271	0.425	0.396	-1.01
3	0.013	0.122	-0.135	0.347	0.463	-1.16
4	0.053	0.245	-0.310	0.539	0.537	-1.37
5	0.105	0.210	-0.351	0.558	0.513	-1.31
6	0.010	0.167	-0.173	0.461	0.598	-1.50
7	0.064	0.412	-0.485	0.765	0.707	-1.81

Table 7.10

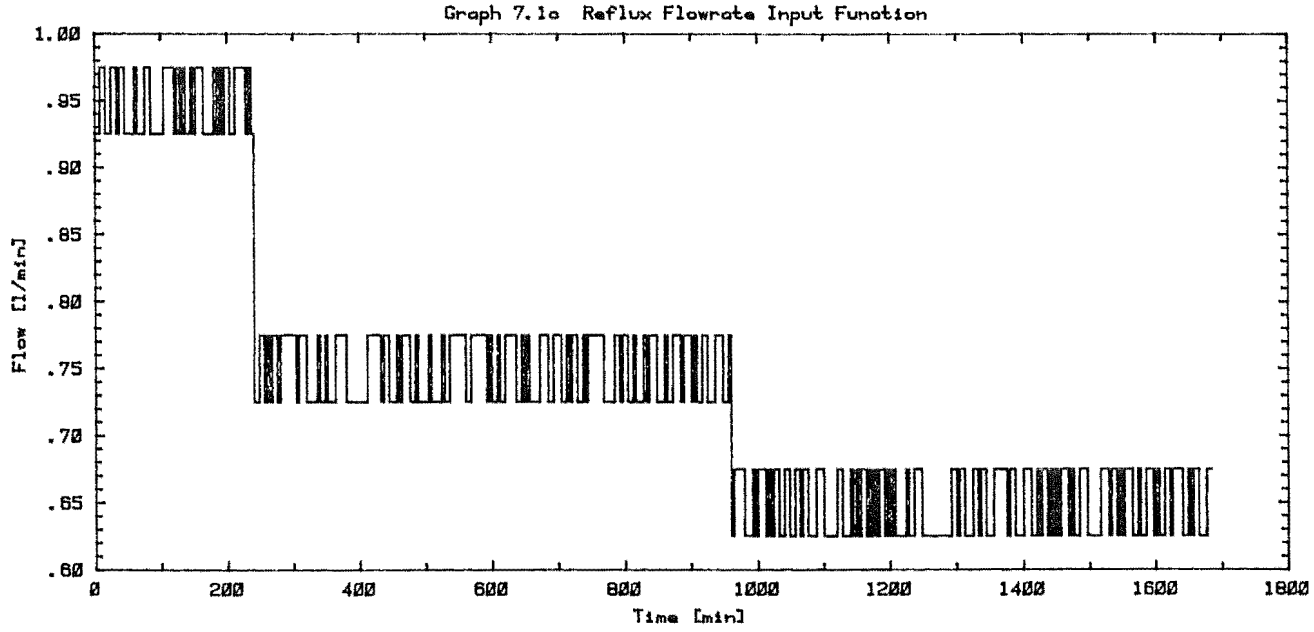
Graph 7.1a Feed Flowrate Input Funotion



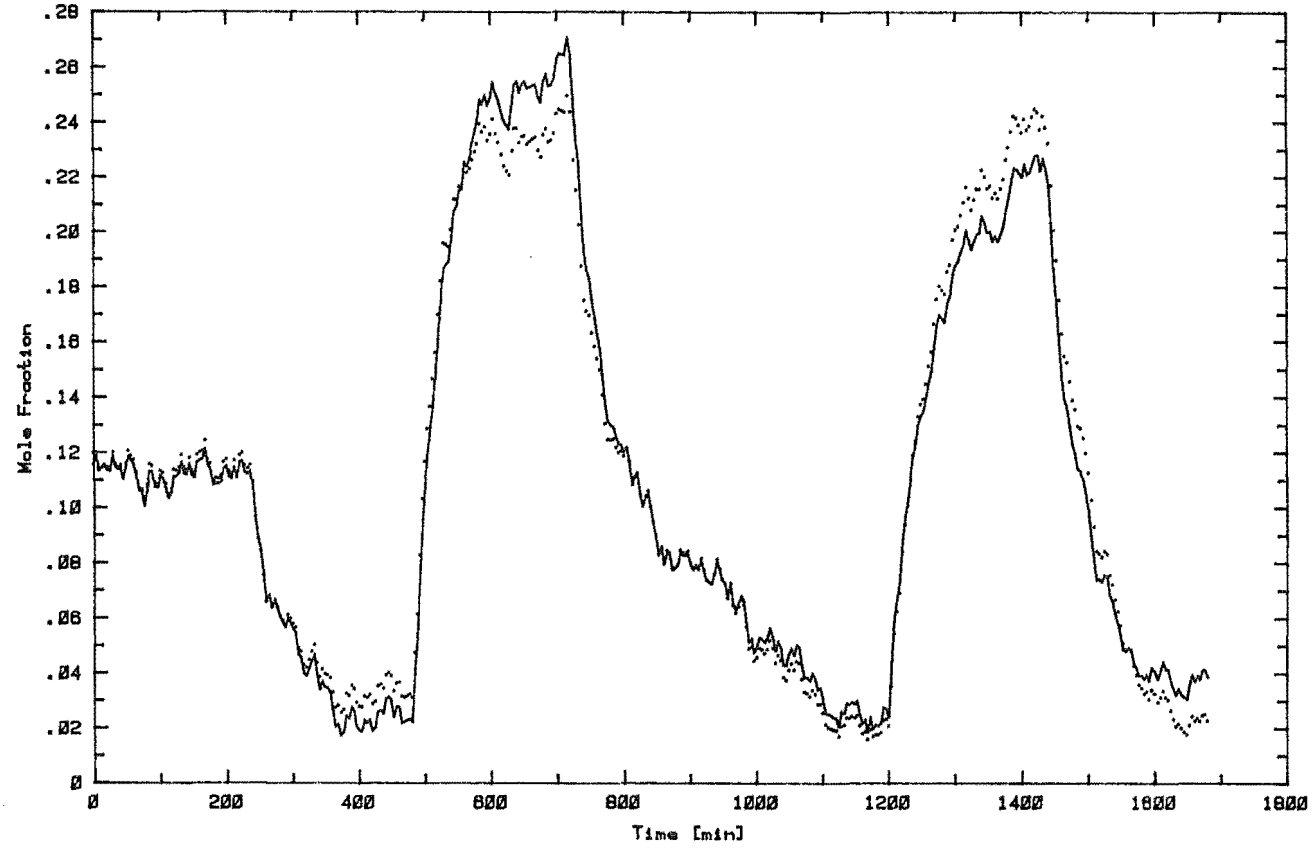
Graph 7.1b Steam Condensate Flowrate Input Funotion



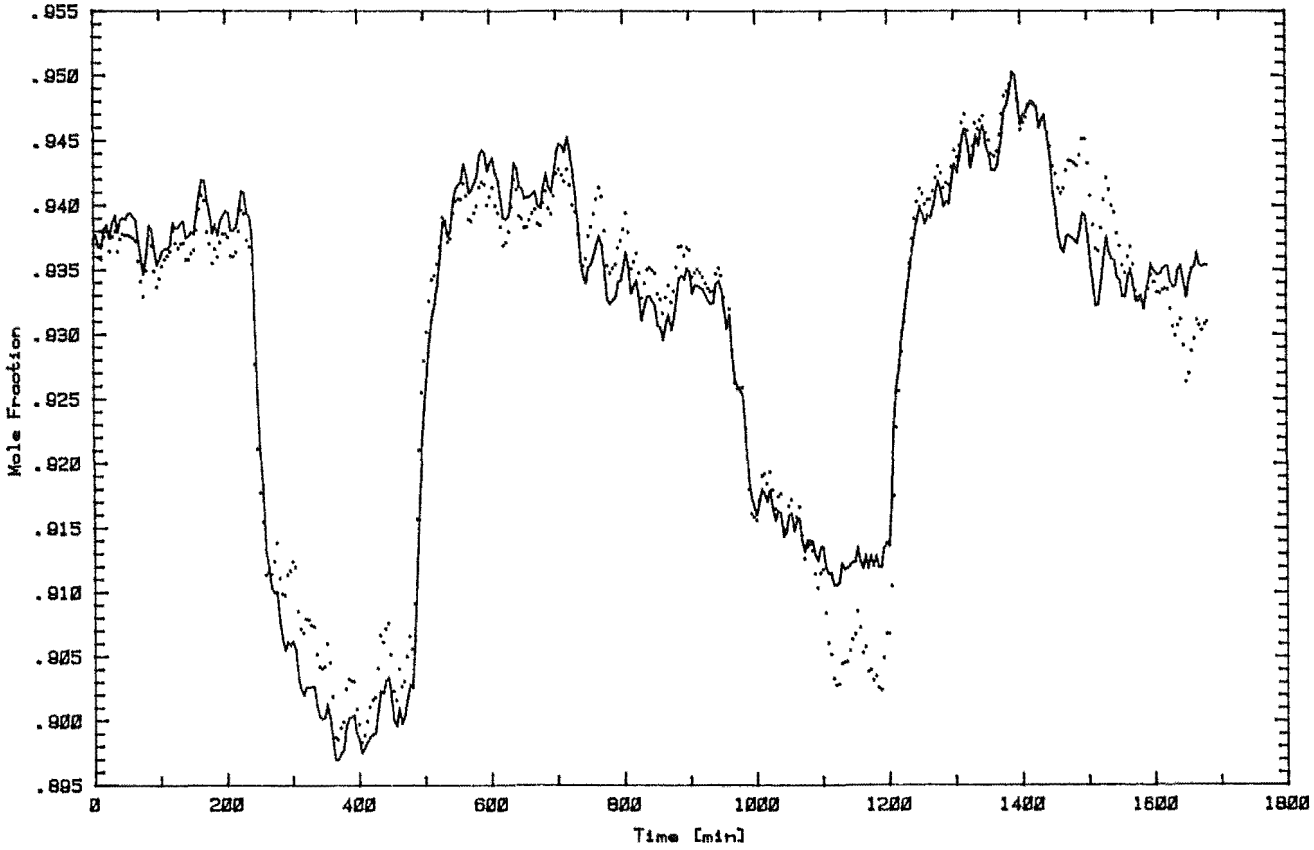
Graph 7.1c Reflux Flowrate Input Funotion



Graph 7.2a 2nd Order Linear Model of Bottome Composition

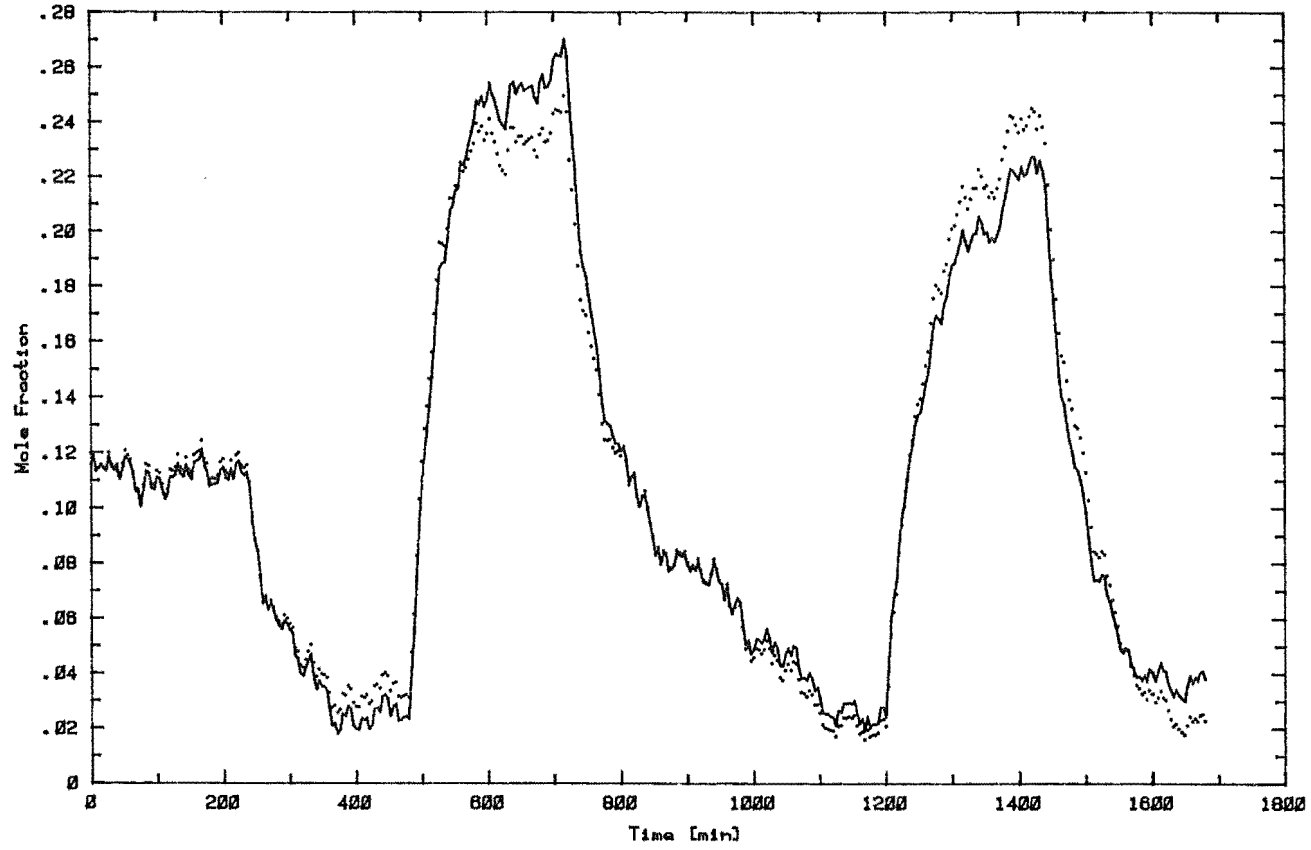


Graph 7.2b 2nd Order Linear Model of Tops Composition

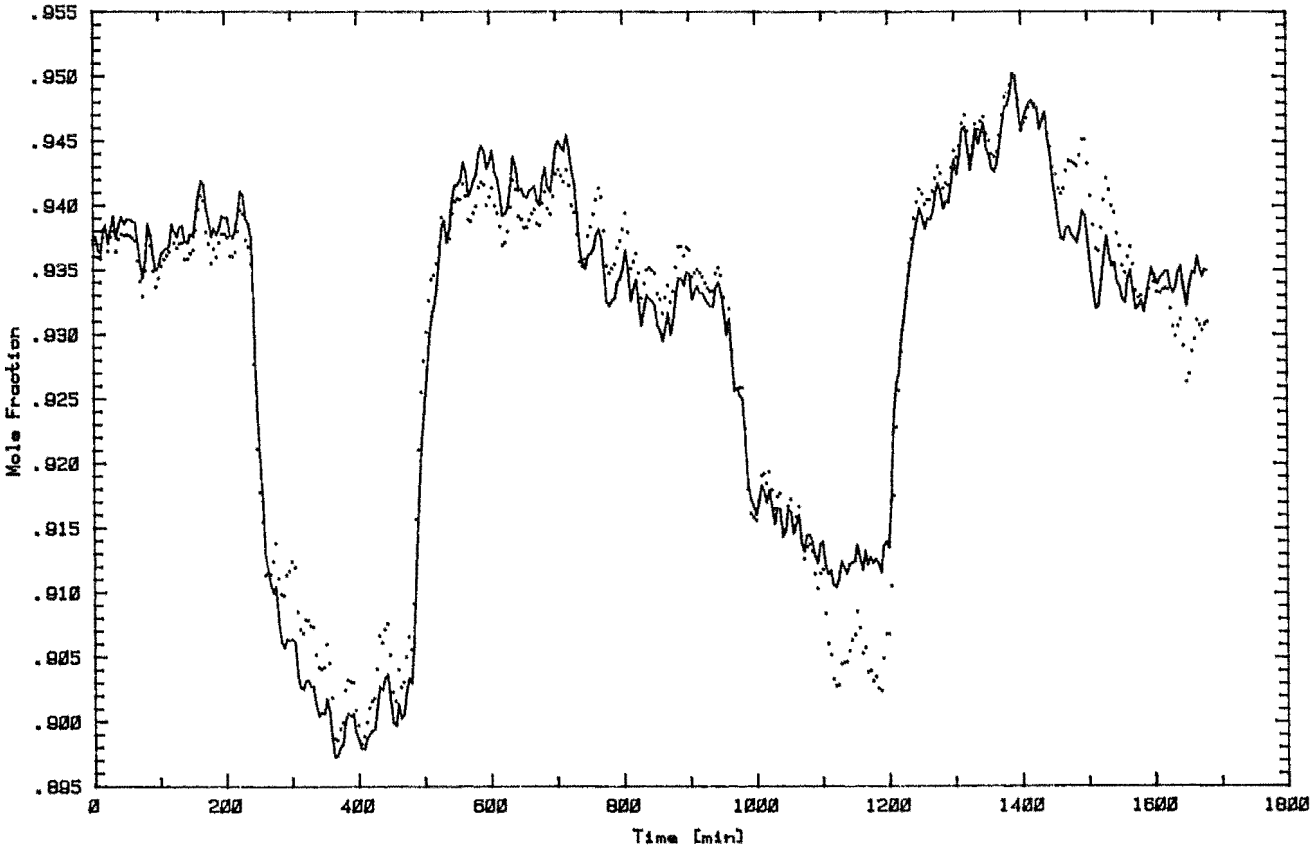


Key: ····· Simulated Column Output
—— Identified Model Output

Graph 7.3a 3rd Order Linear Model of Bottome Composition

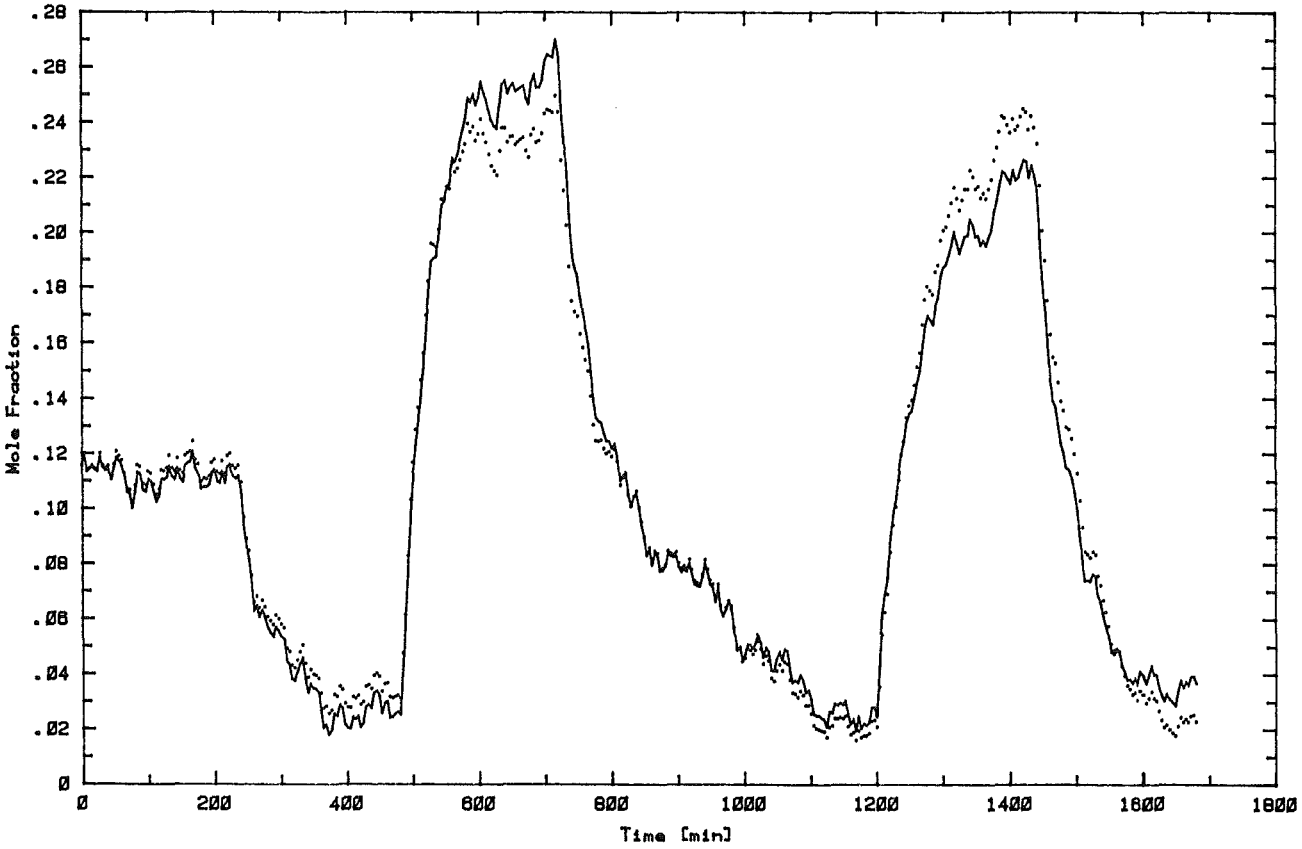


Graph 7.3b 3rd Order Linear Model of Tops Composition

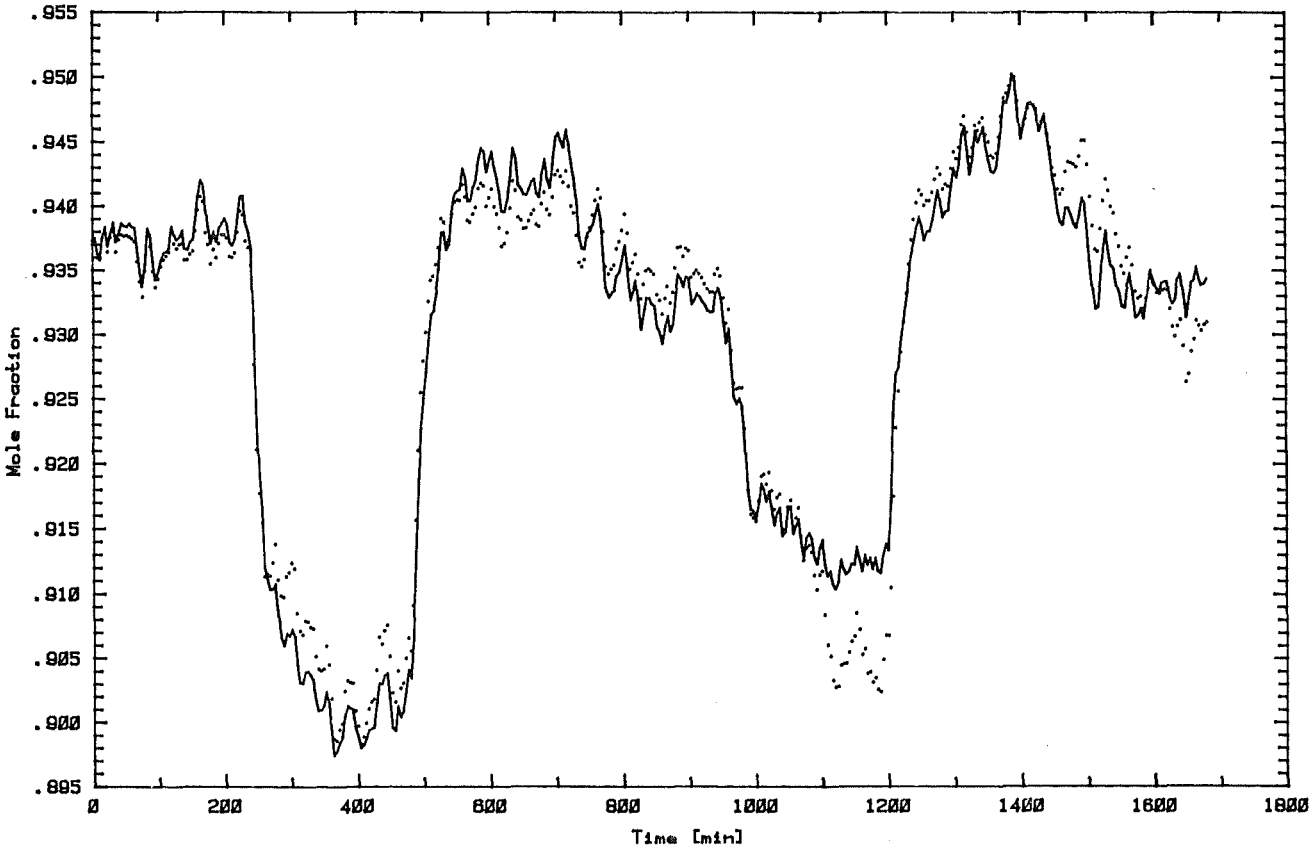


Key: ····· Simulated Column Output
—— Identified Model Output

Graph 7.4a 4th Order Linear Model of Bottome Composition

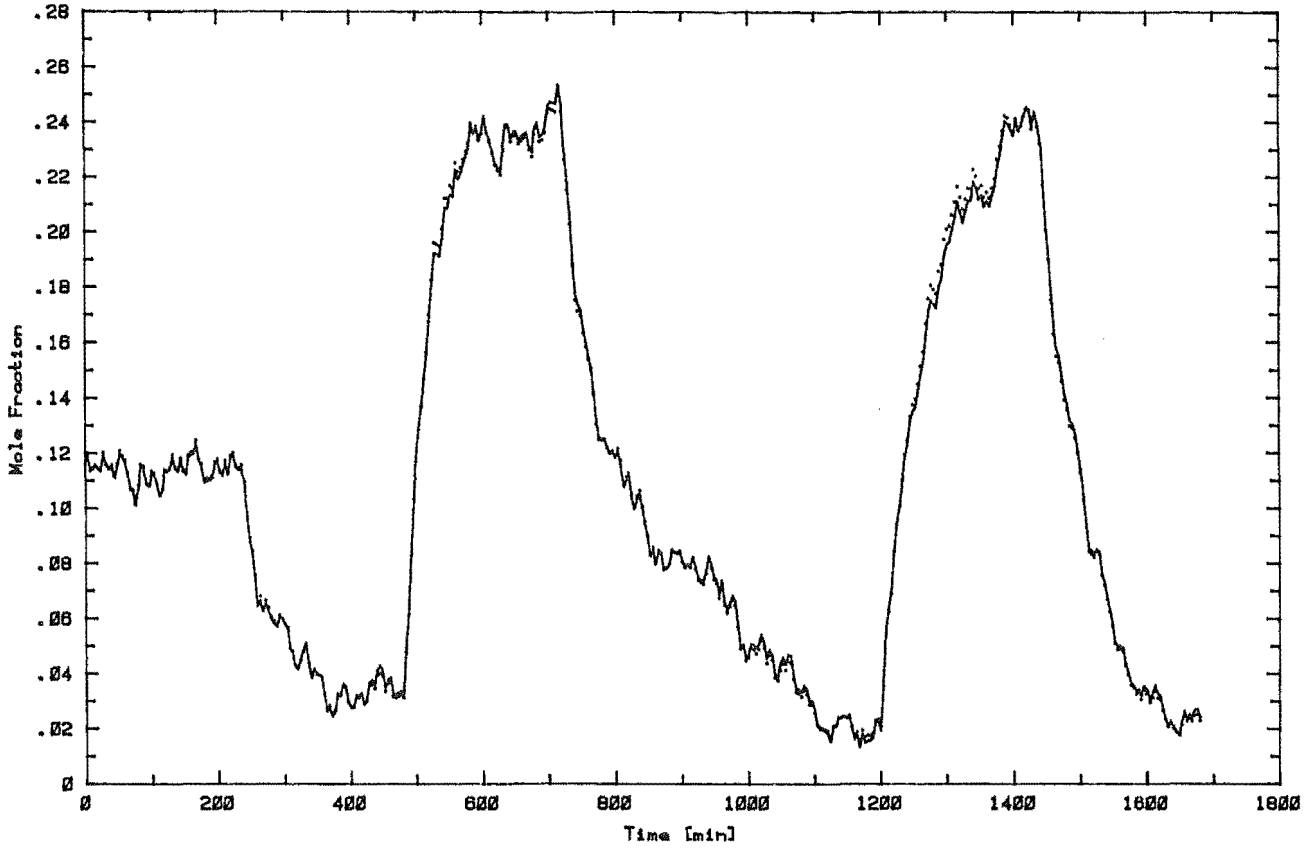


Graph 7.4b 4th Order Linear Model of Tops Composition

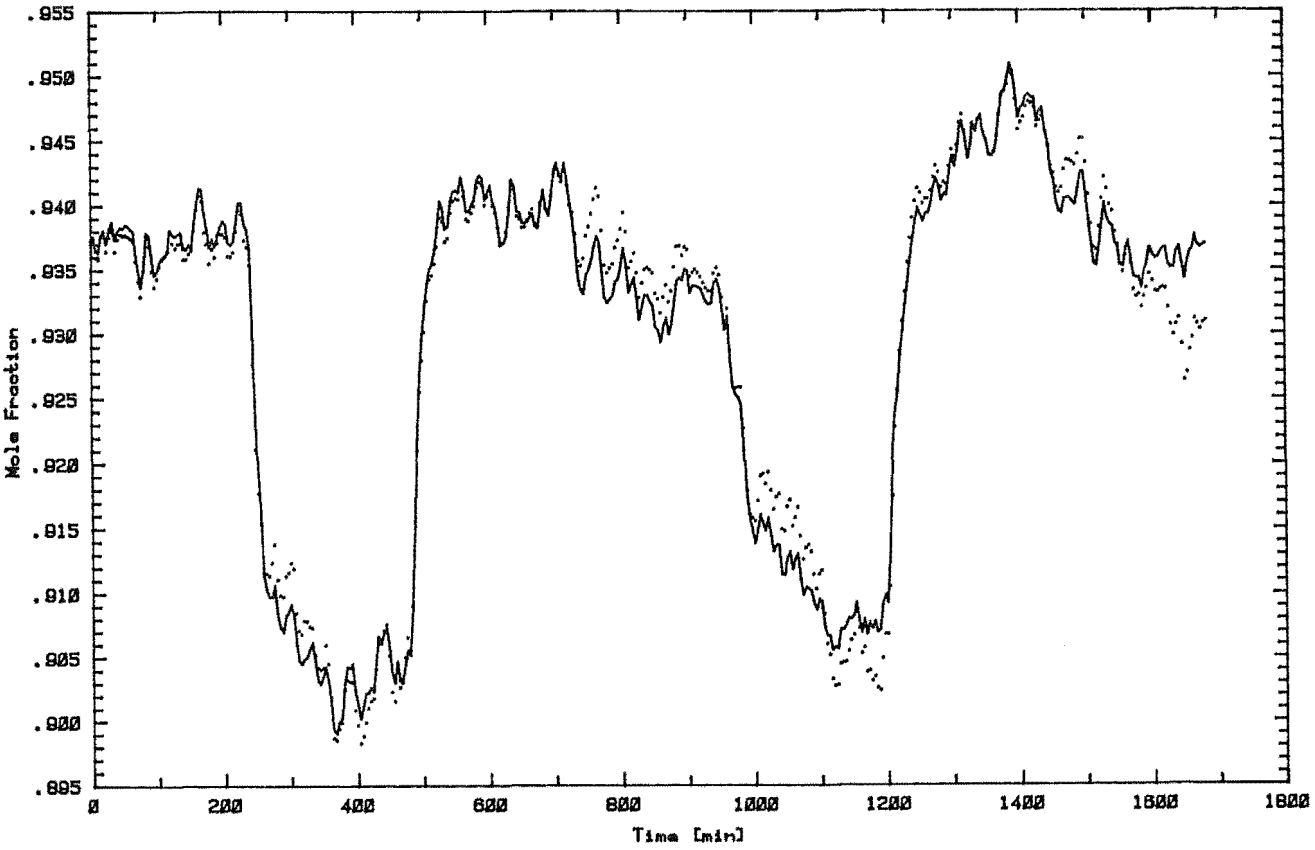


Key: ····· Simulated Column Output
—— Identified Model Output

Graph 7.5a 2nd Order Diagonal Bilinear Model of Bottome Compoeition

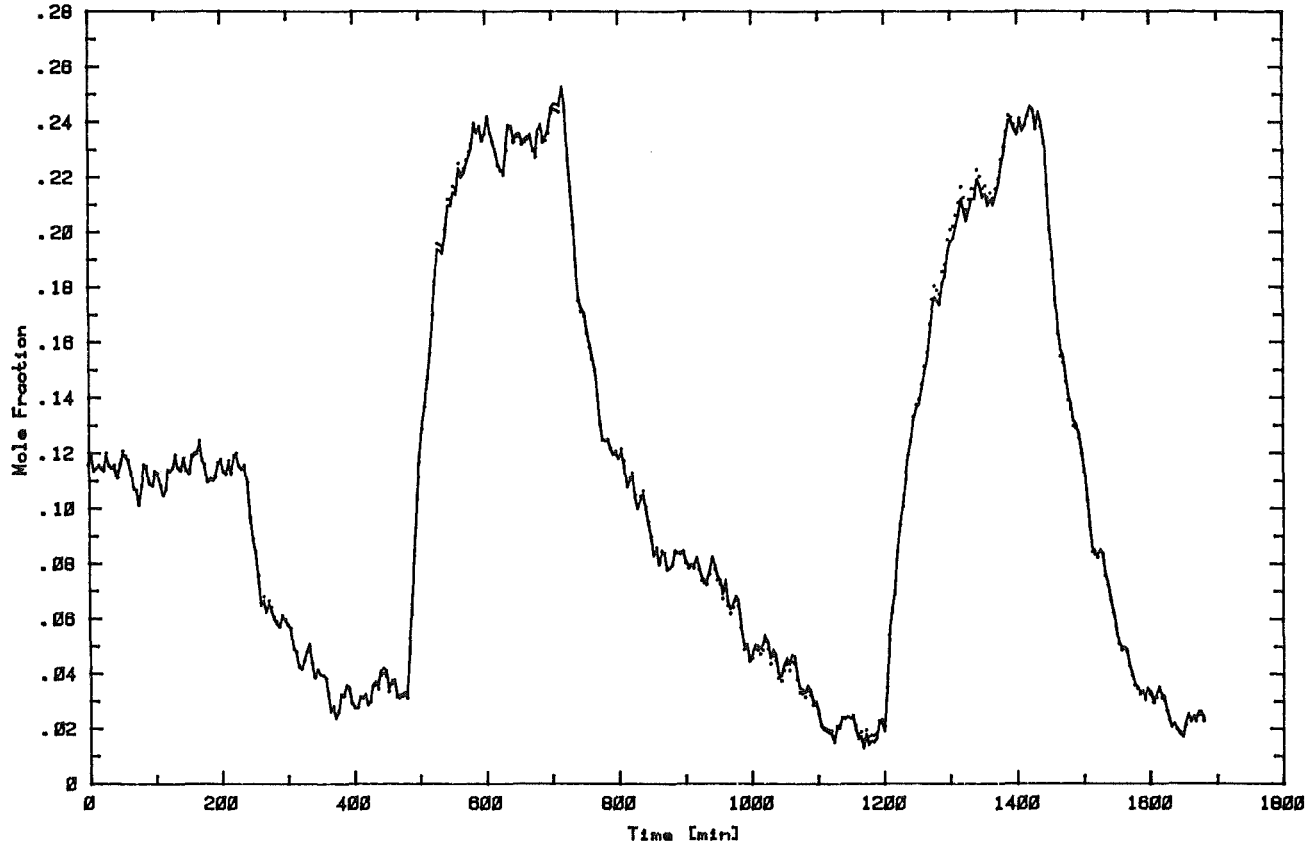


Graph 7.5b 2nd Order Diagonal Bilinear Model of Tops Compoeition

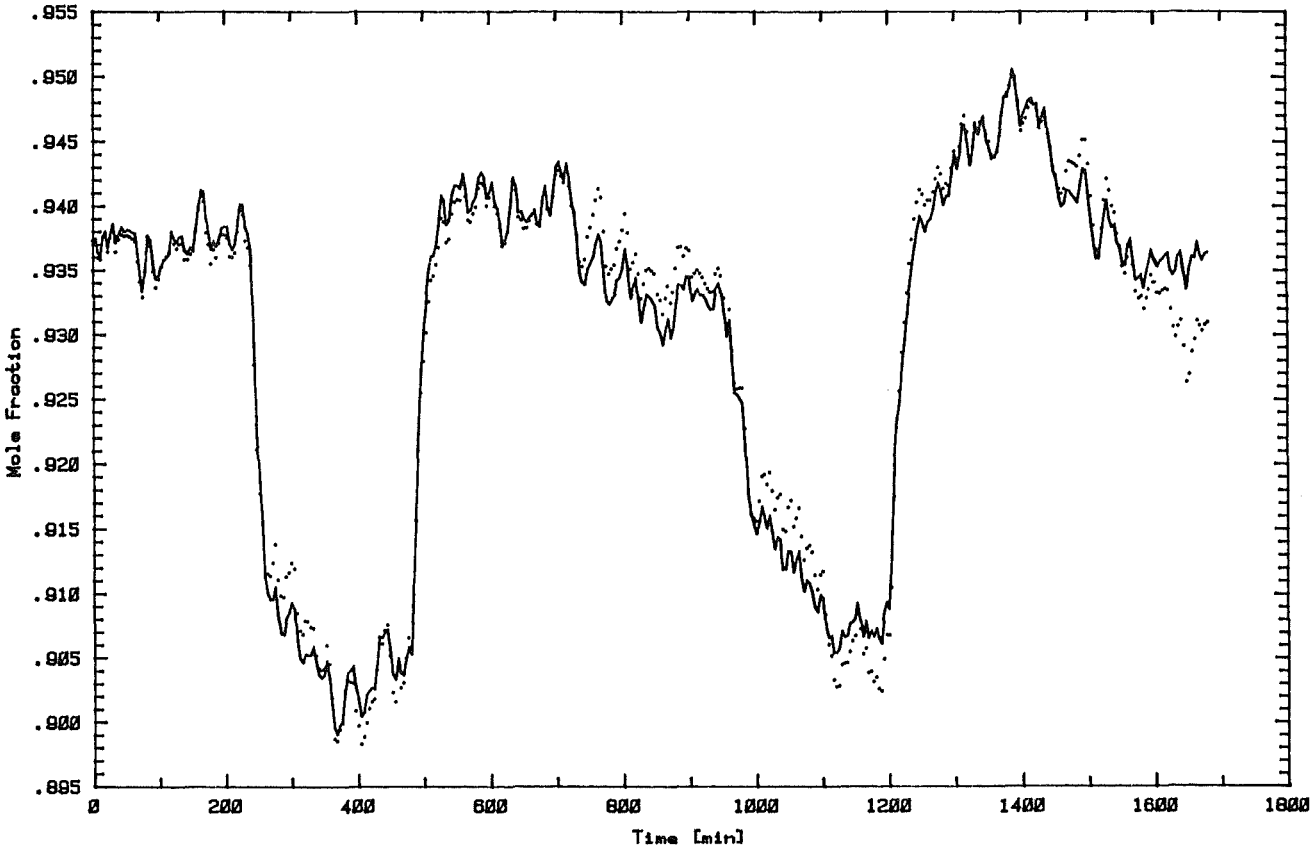


Key: ····· Simulated Column Output
—— Identified Model Output

Graph 7.6a 3rd Order Diagonal Bilinear Model of Bottome Composition

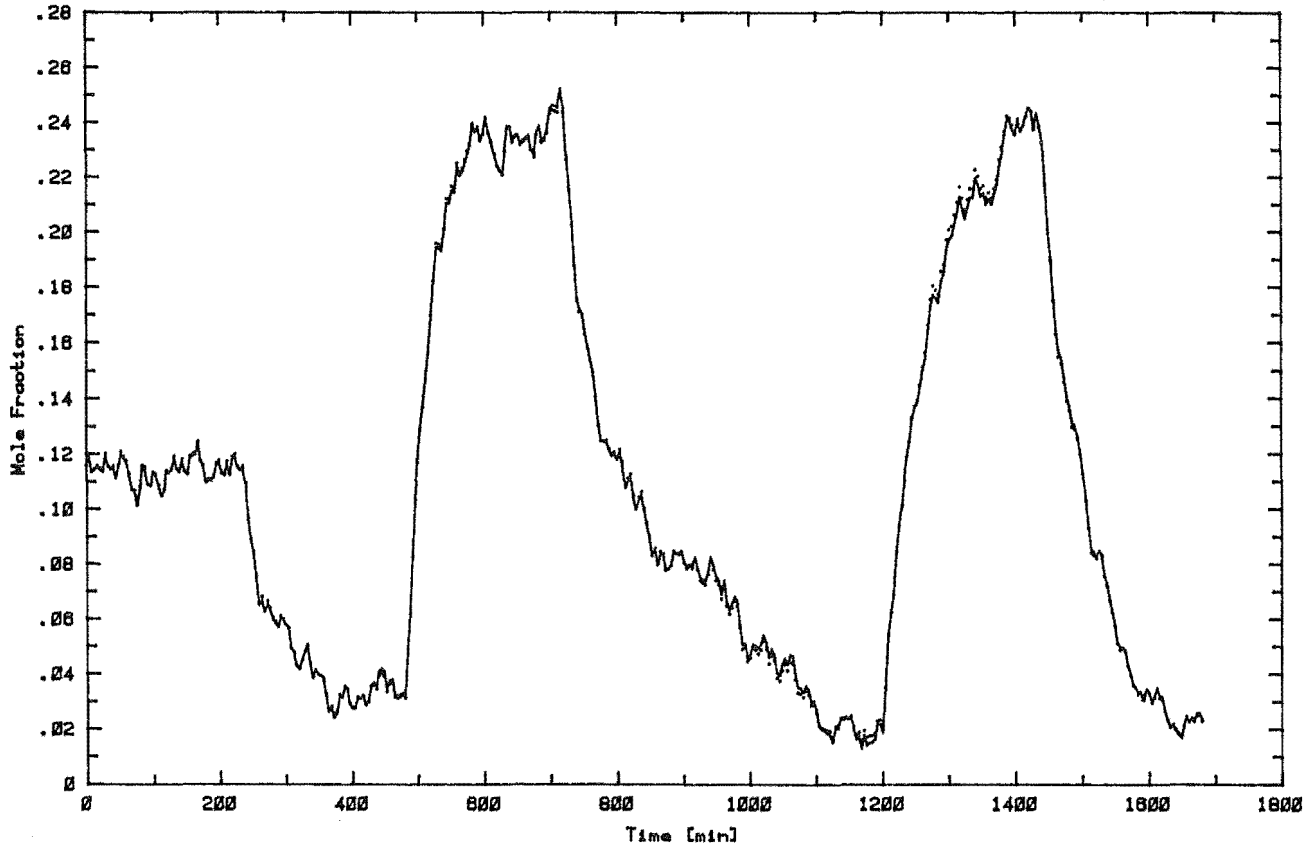


Graph 7.6b 3rd Order Diagonal Bilinear Model of Tops Composition

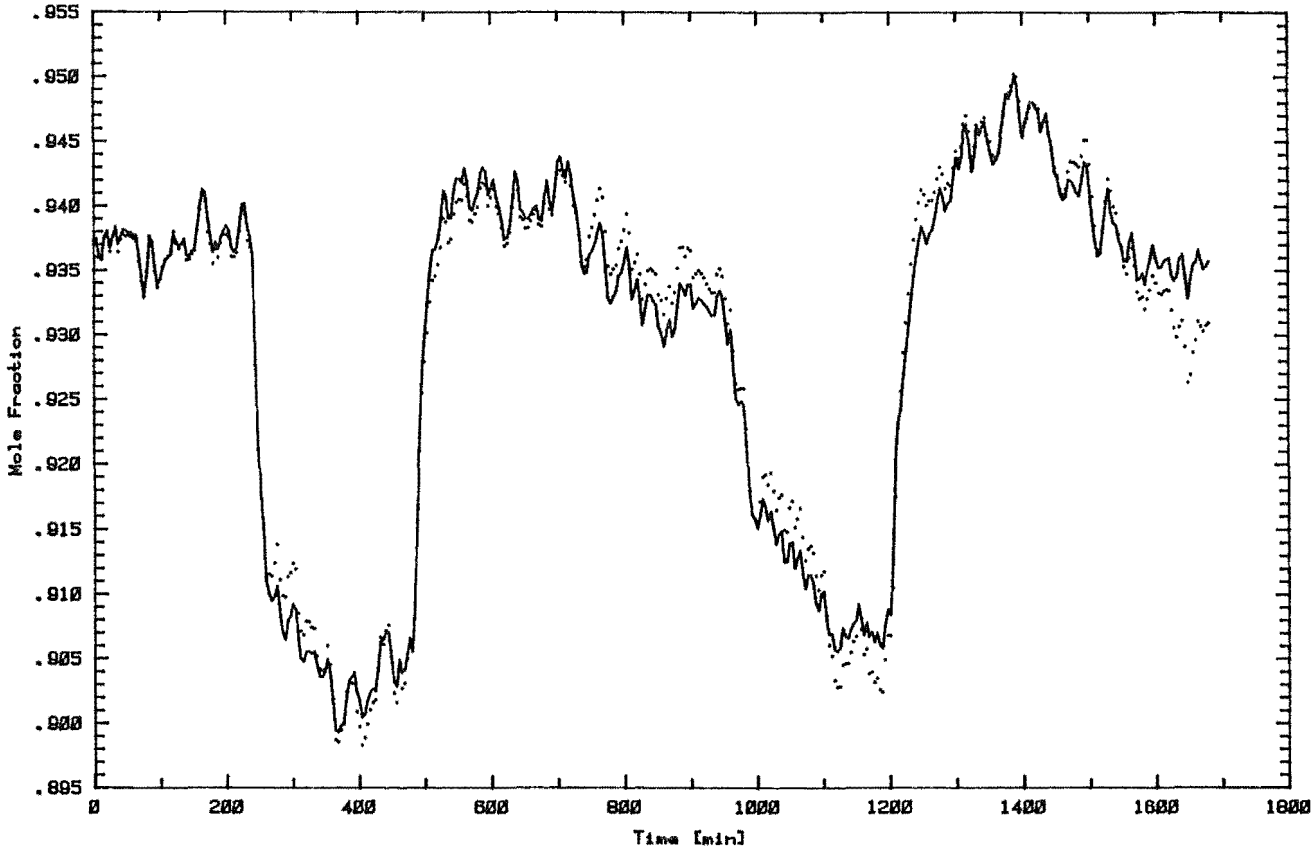


Keys: ····· Simulated Column Output
—— Identified Model Output

Graph 7.7a 4th Order Diagonal Bilinear Model of Bottoms Composition

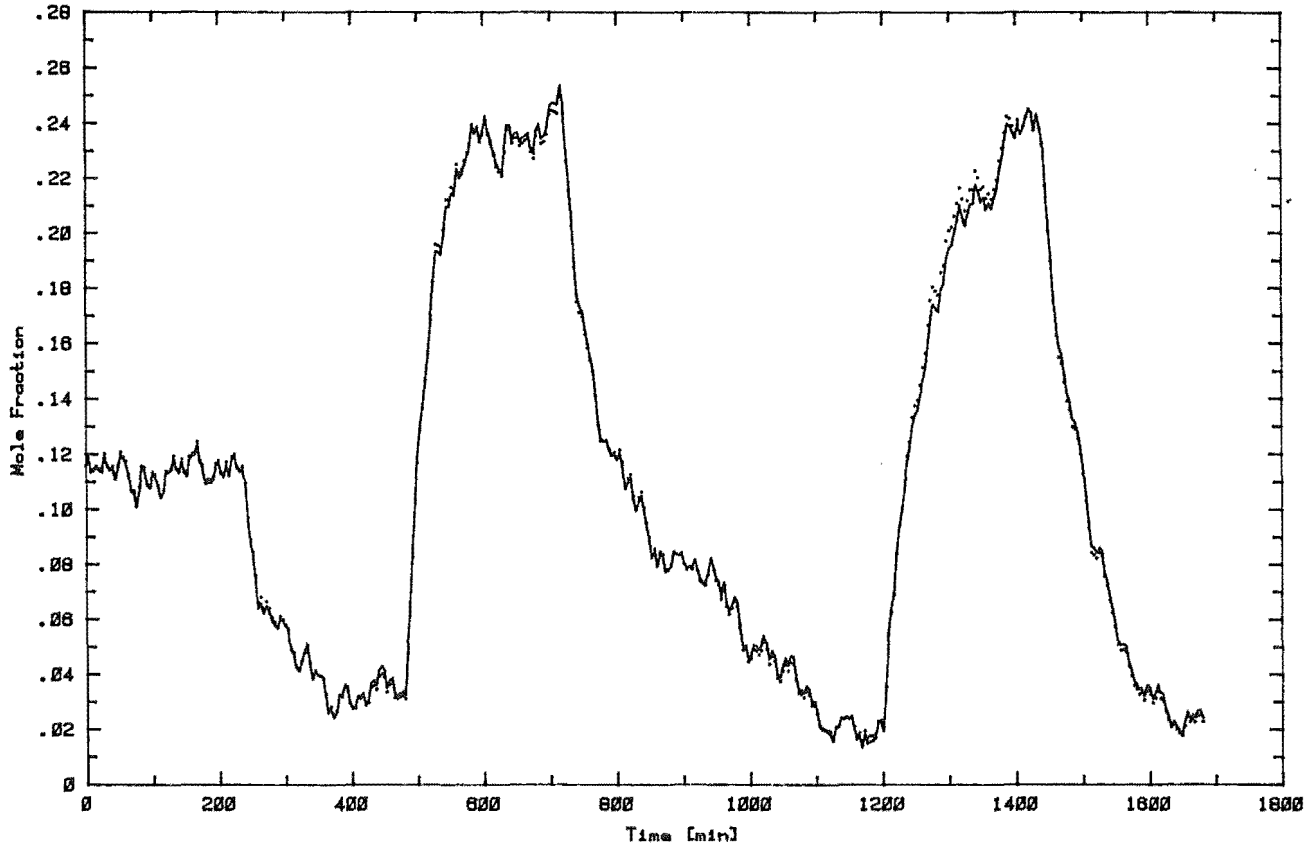


Graph 7.7b 4th Order Diagonal Bilinear Model of Tops Composition

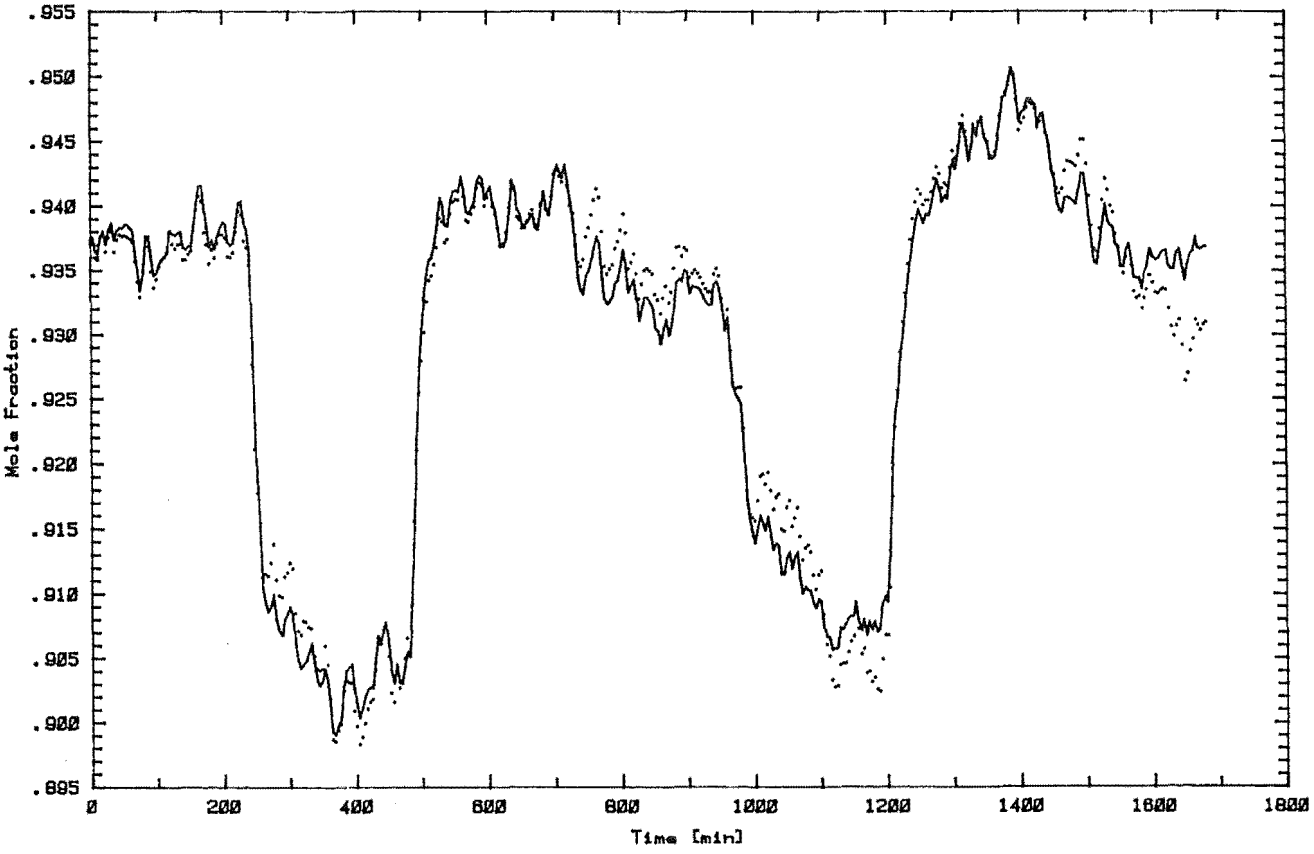


Key: ····· Simulated Column Output
—— Identified Model Output

Graph 7.8a 2nd Order Bilinear Model of Bottome Composition

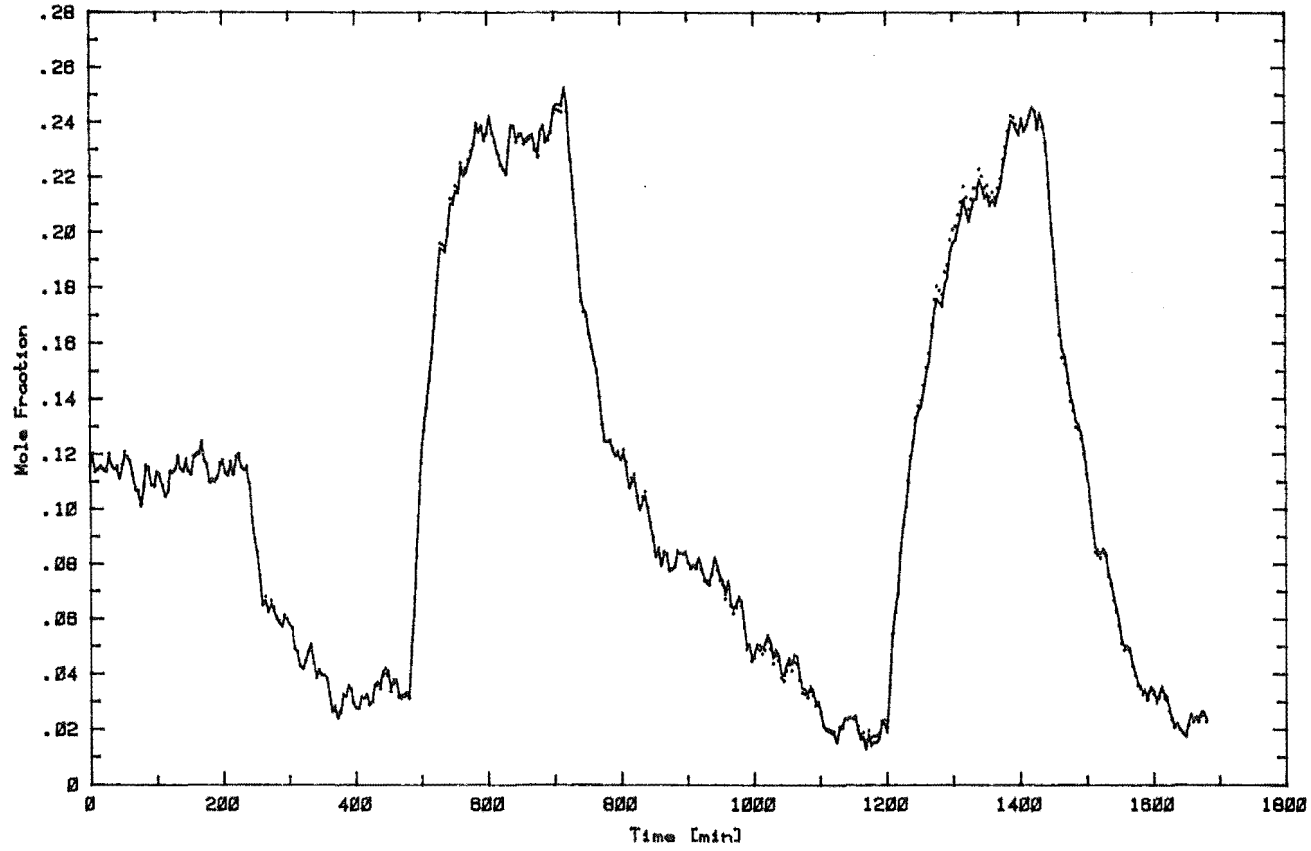


Graph 7.8b 2nd Order Bilinear Model of Tops Composition

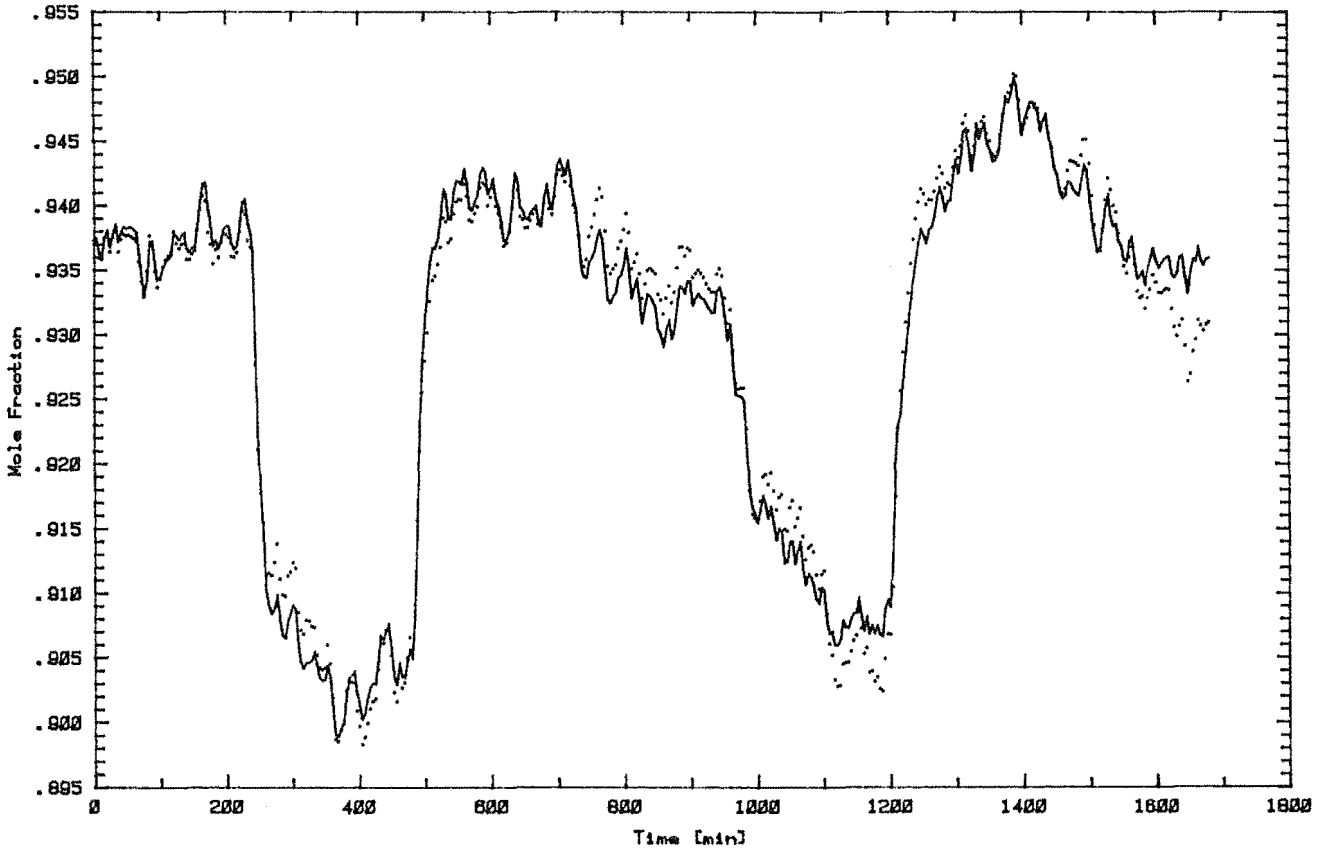


Key: ····· Simulated Column Output
—— Identified Model Output

Graph 7.9a 3rd Order Bilinear Model of Bottome Composition

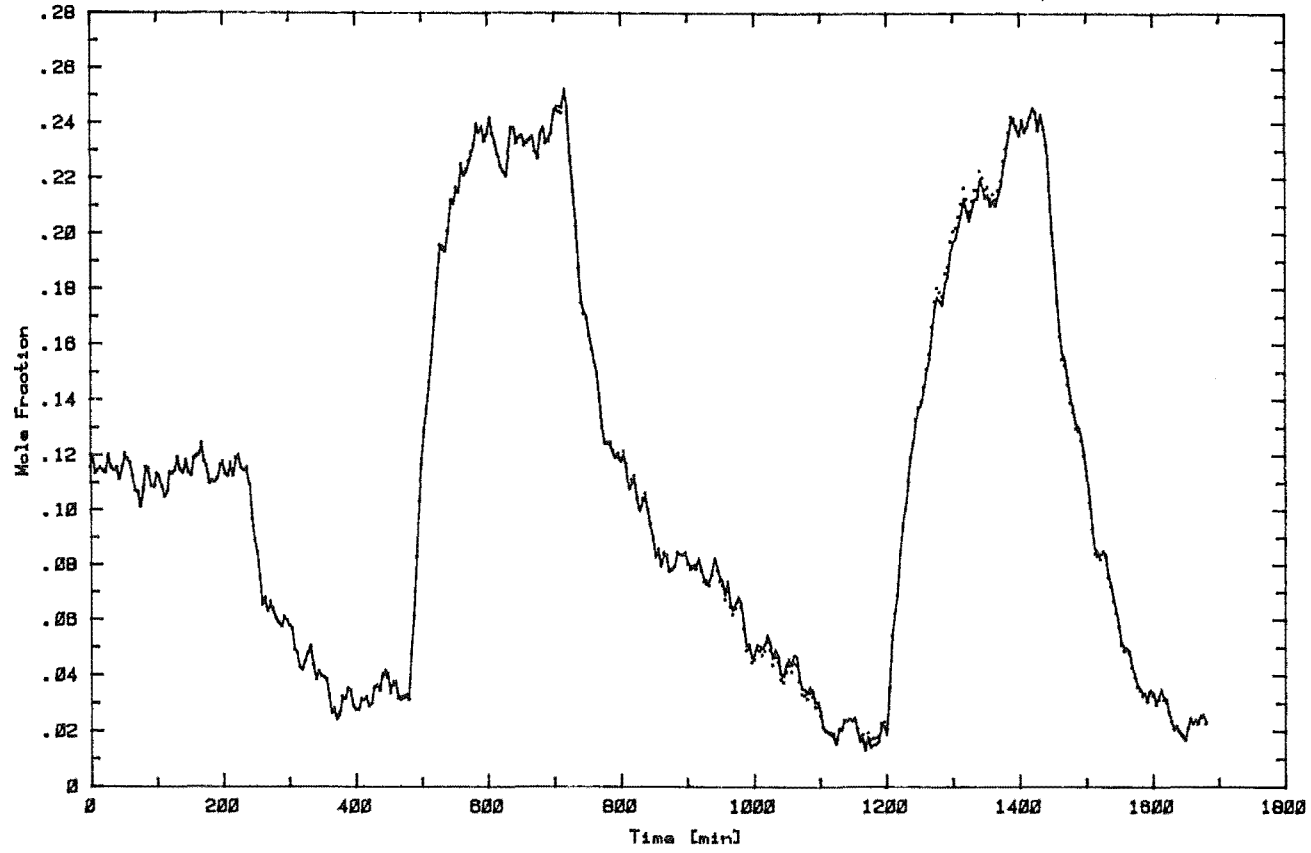


Graph 7.9b 3rd Order Bilinear Model of Tope Composition

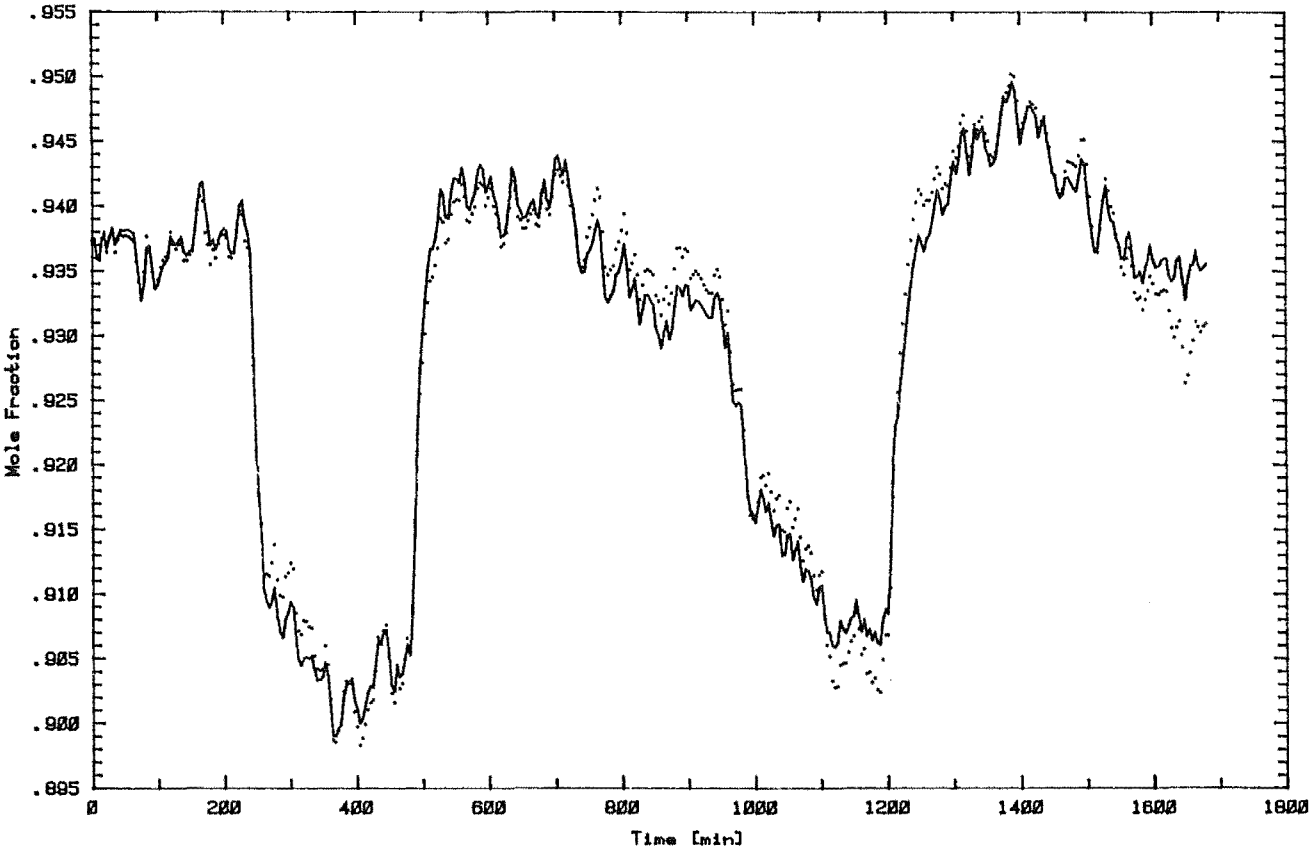


Key: ····· Simulated Column Output
—— Identified Model Output

Graph 7.10a 4th Order Bilinear Model of Bottome Composition



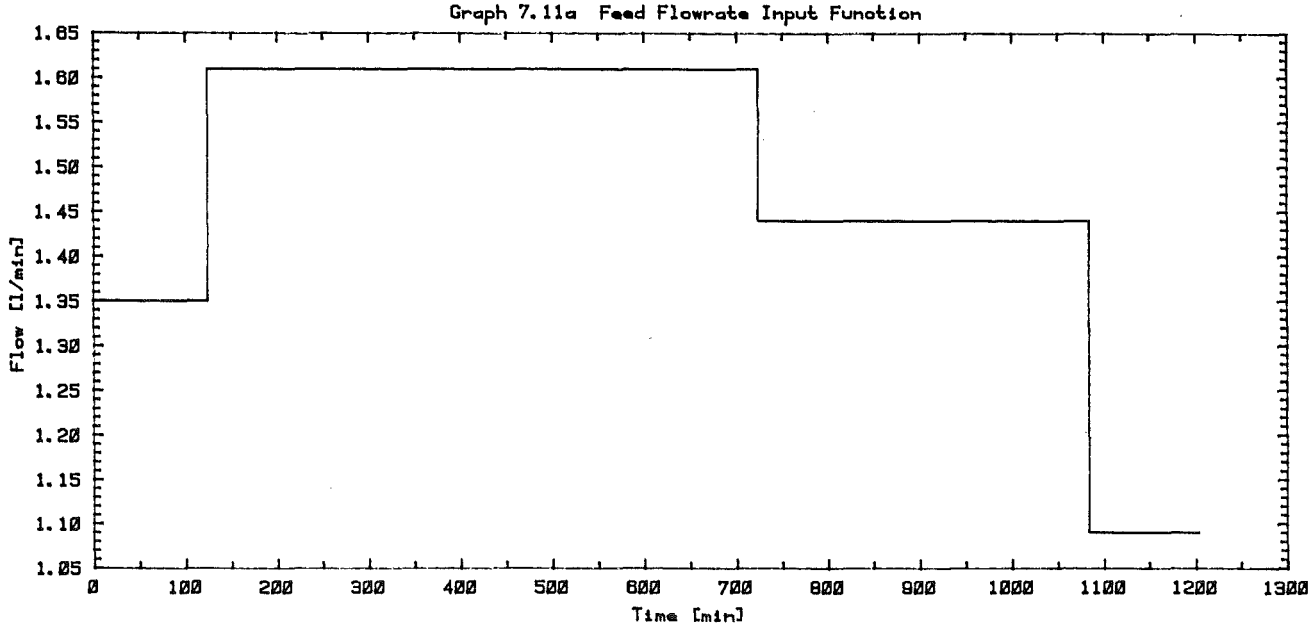
Graph 7.10b 4th Order Bilinear Model of Tops Composition



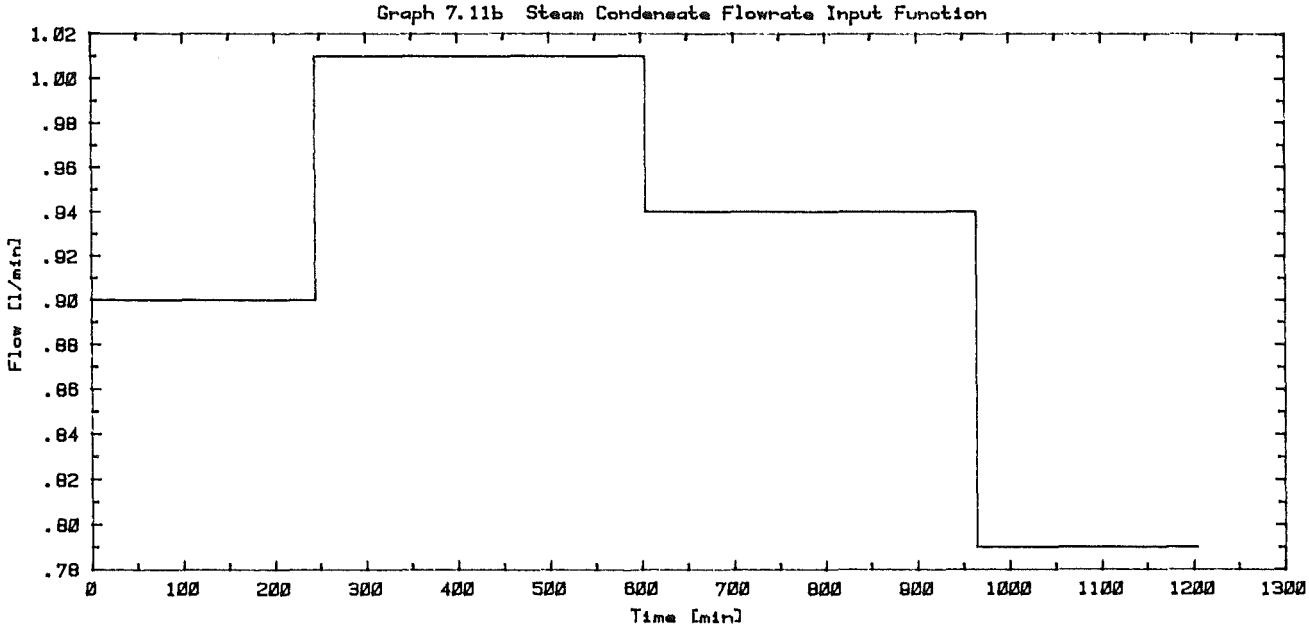
Key: ····· Simulated Column Output
—— Identified Model Output

In order to test the ability of the identified models to predict the future behaviour of the simulated column, the fourth order models identified using input/output data generated with the input function shown in Graph 7.1 were used to predict the output corresponding to the input functions shown in Graph 7.11. The simulated outputs and the predicted outputs for the different types of models are shown in Graphs 7.12 to 7.14. These graphs show that all the model types tried gave good results for predicting the bottoms composition, but poor results for the tops composition.

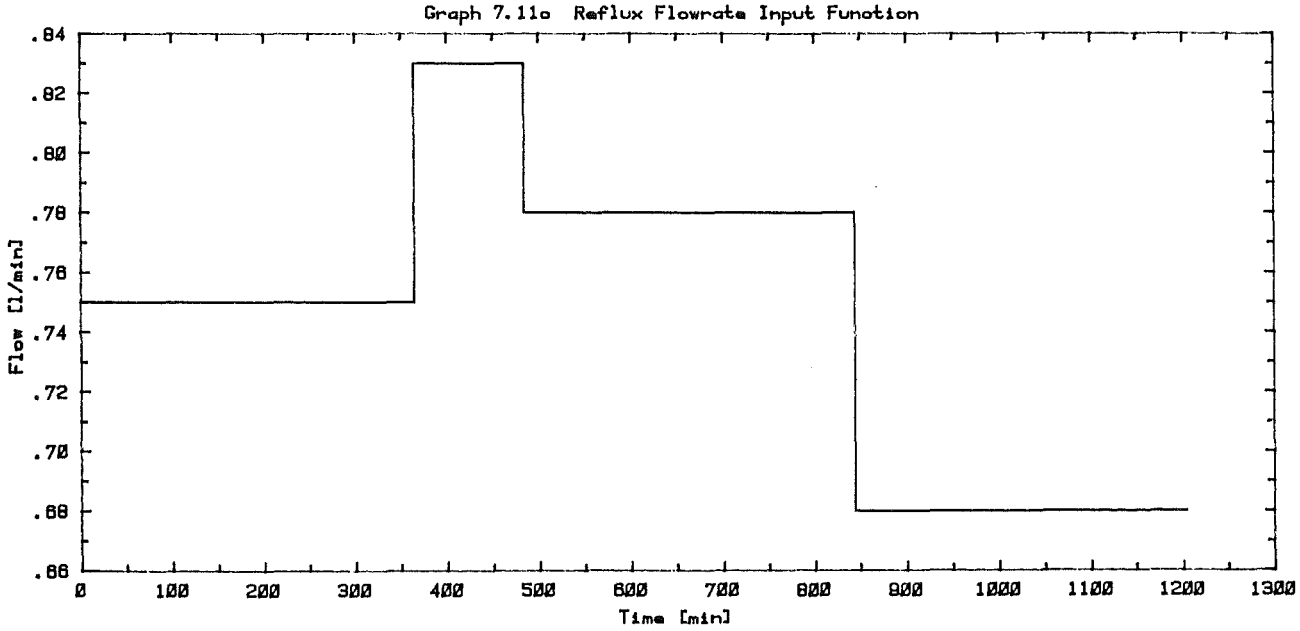
Graph 7.11a Feed Flowrate Input Function



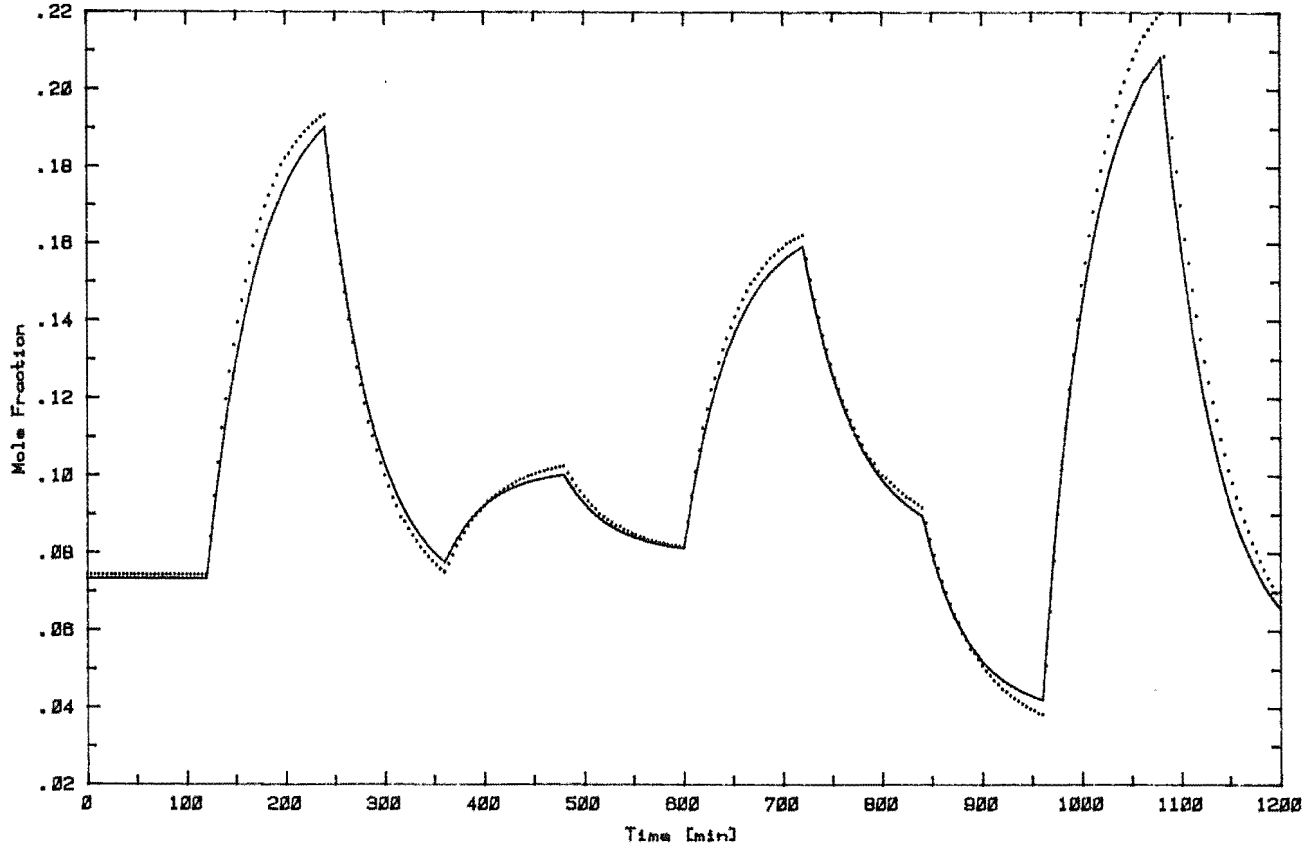
Graph 7.11b Steam Condensate Flowrate Input Function



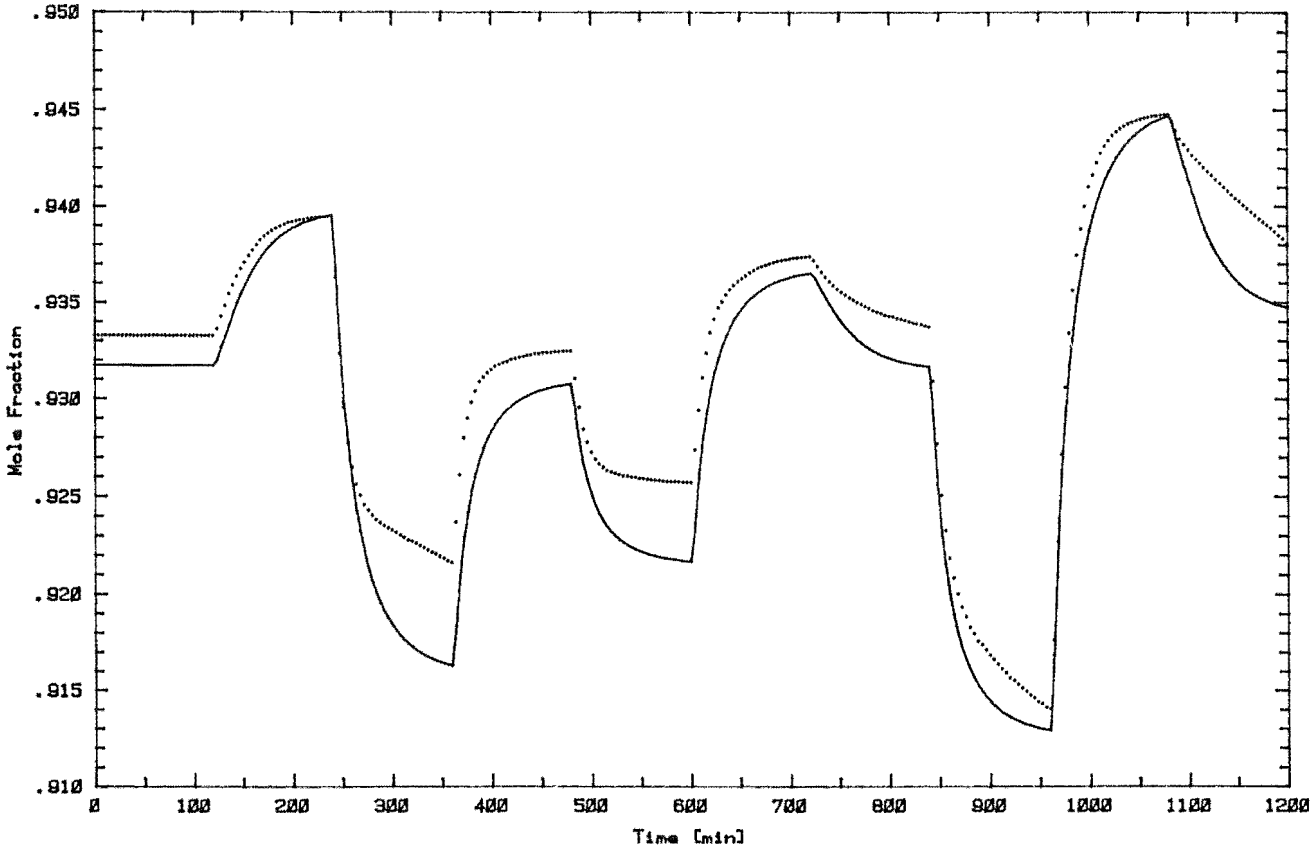
Graph 7.11c Reflux Flowrate Input Function



Graph 7.12a 4th Order Linear Model of Predicted Bottoms Composition

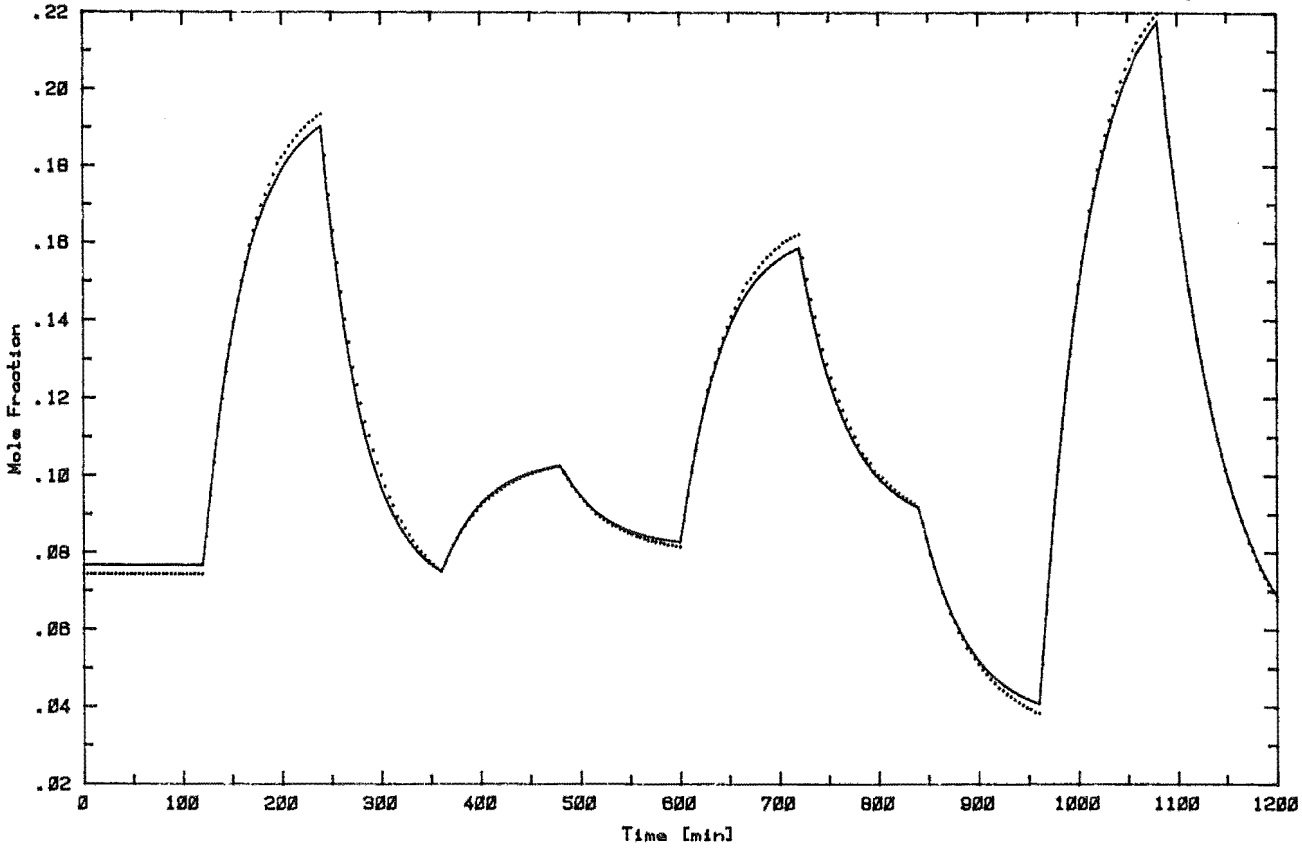


Graph 7.12b 4th Order Linear Model of Predicted Tops Composition

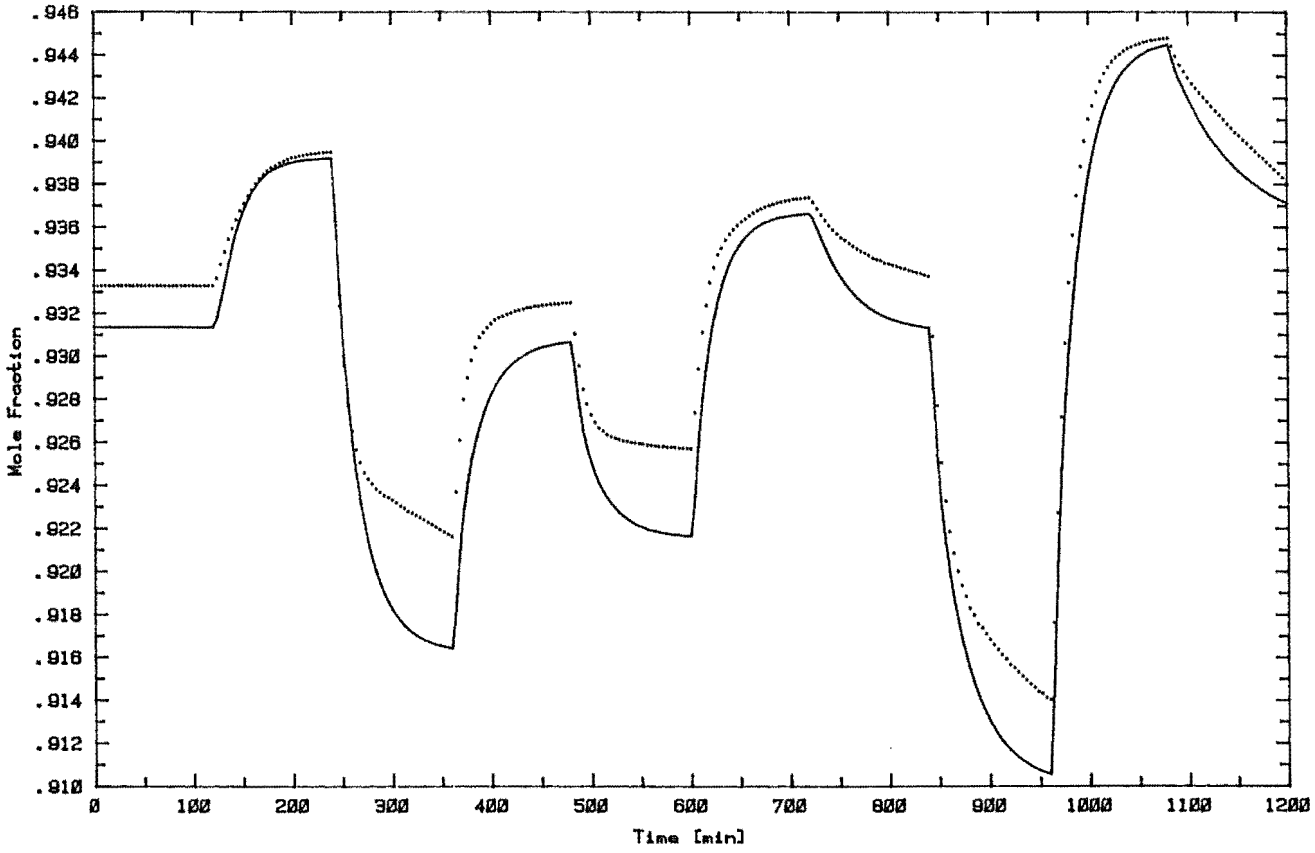


Key: ····· Simulated Column Output
—— Identified Model Output

Graph 7.13a 4th Order Diagonal Bilinear Model of Predicted Bottome Composition

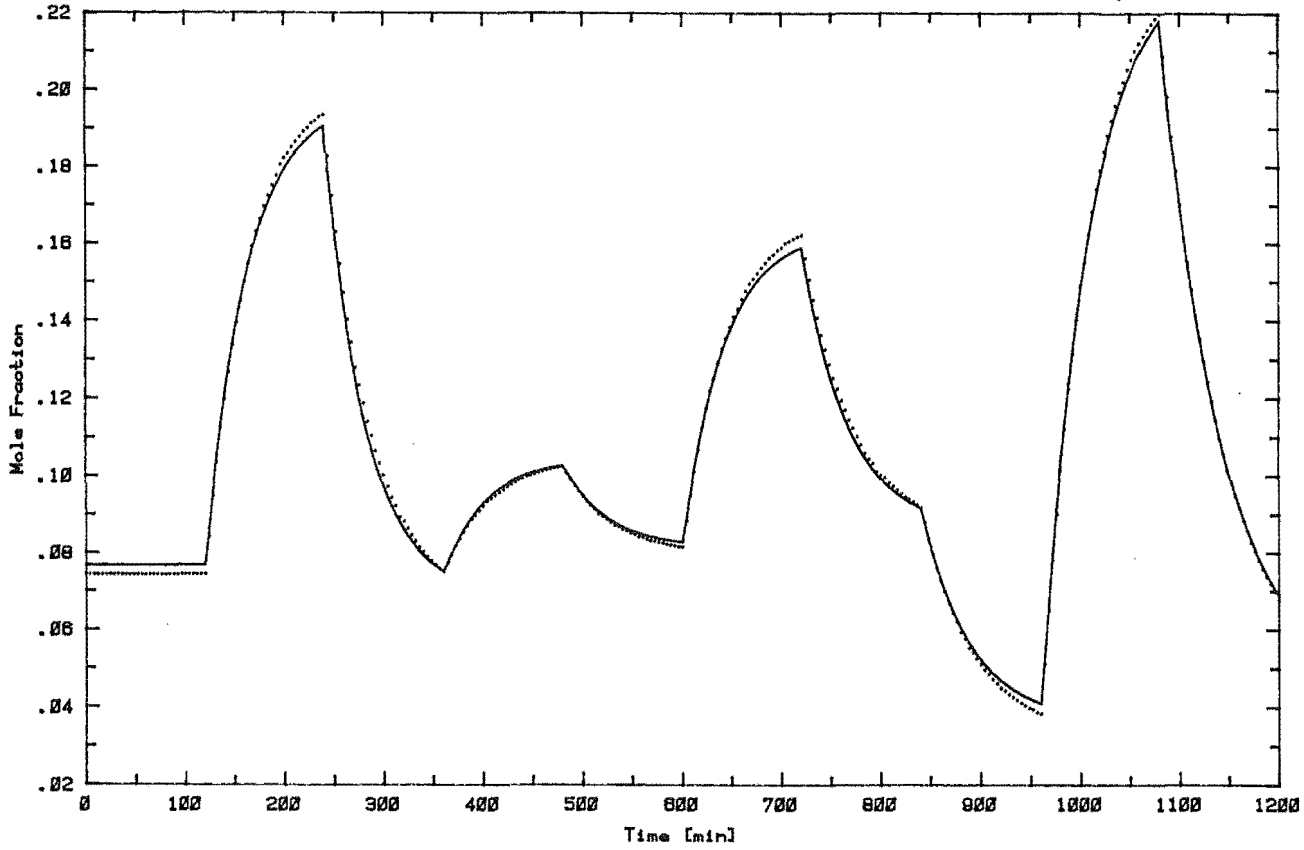


Graph 7.13b 4th Order Diagonal Bilinear Model of Predicted Tops Composition

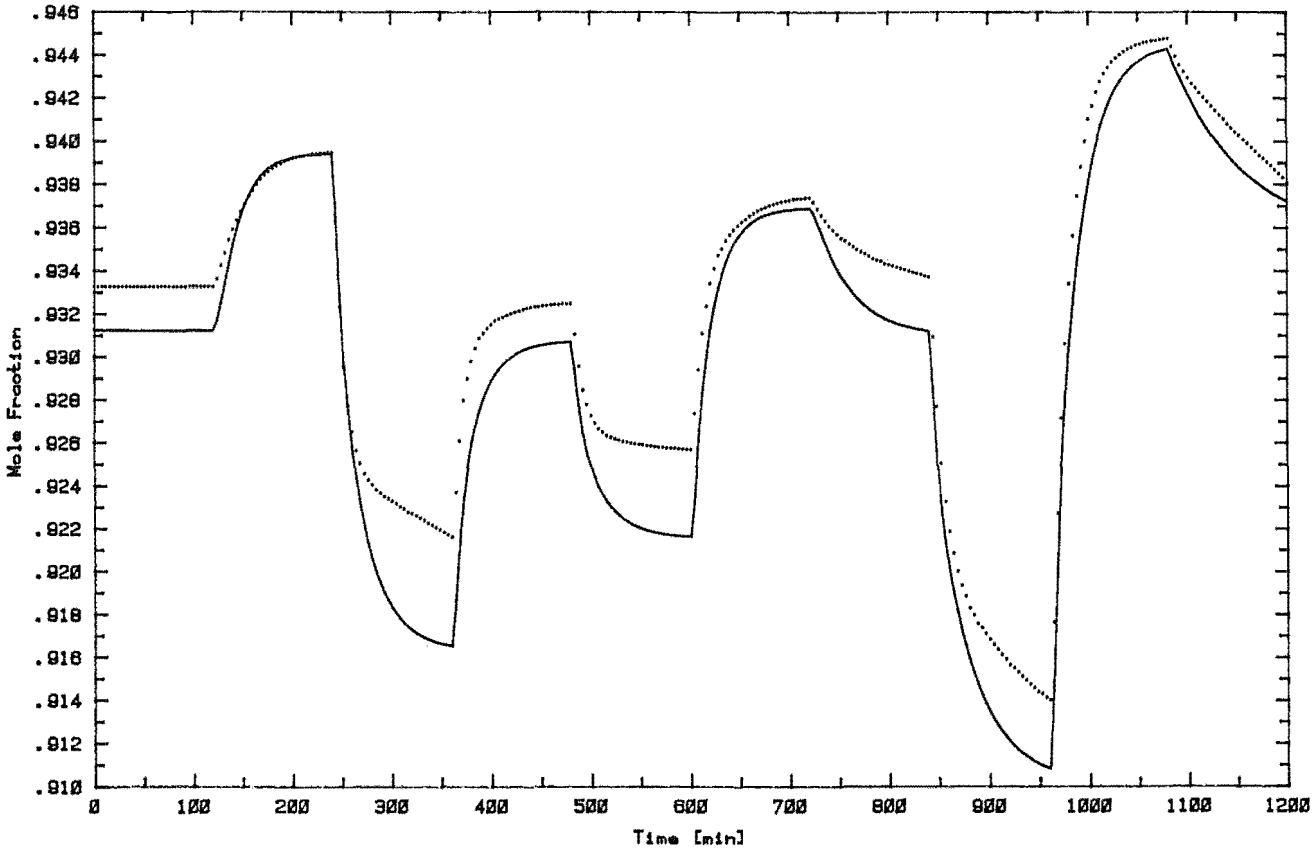


Key: ····· Simulated Column Output
—— Identified Model Output

Graph 7.14a 4th Order Bilinear Model of Predicted Bottome Compoeition



Graph 7.14b 4th Order Bilinear Model of Predicted Tope Compoeition



Key: ····· Simulated Column Output
—— Identified Model Output

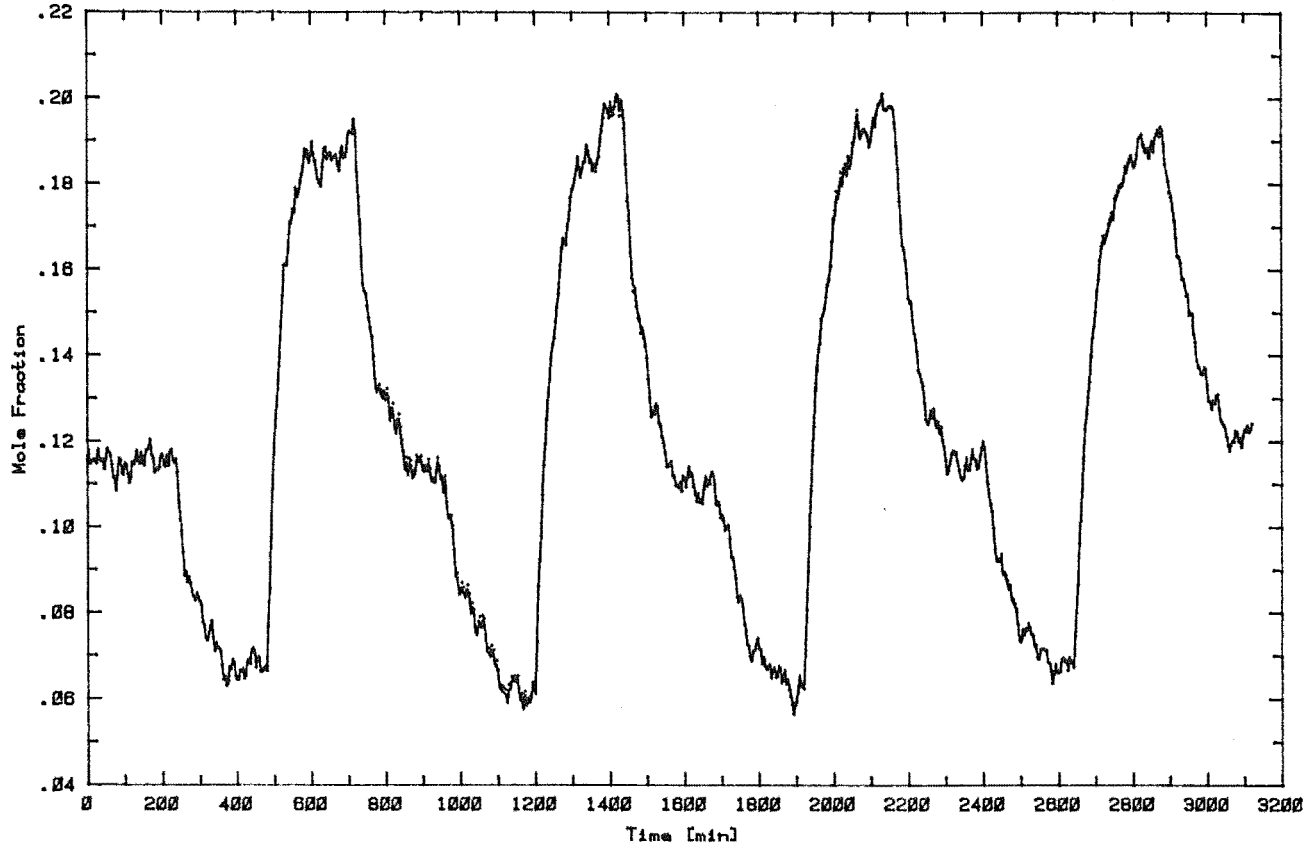
As mentioned in Section 7.1, the expected validity region for the bilinear model was a truncated cone with its apex at the origin. To test this assertion another input function was used. This consisted of stepping between thirteen operating points shown in Appendix F.1 . This function drove the column over an operating region with the same turn-down ratio as that shown in Graph 7.1, but with the combinations of input variables chosen to give smaller variations in column compositions. The simulated and the identified model outputs for a fourth order diagonal bilinear model are shown in Graph 7.15. The variances of the differences between the simulated outputs and the identified model outputs for several different model types are shown in Table 7.11. These are between five and twenty times smaller than the variances shown in Table 7.3 for the original input function. This implies that the new operating region more accurately reflects the validity region of different models used.

Variances of Model Errors
for Input Function in Appendix F.1

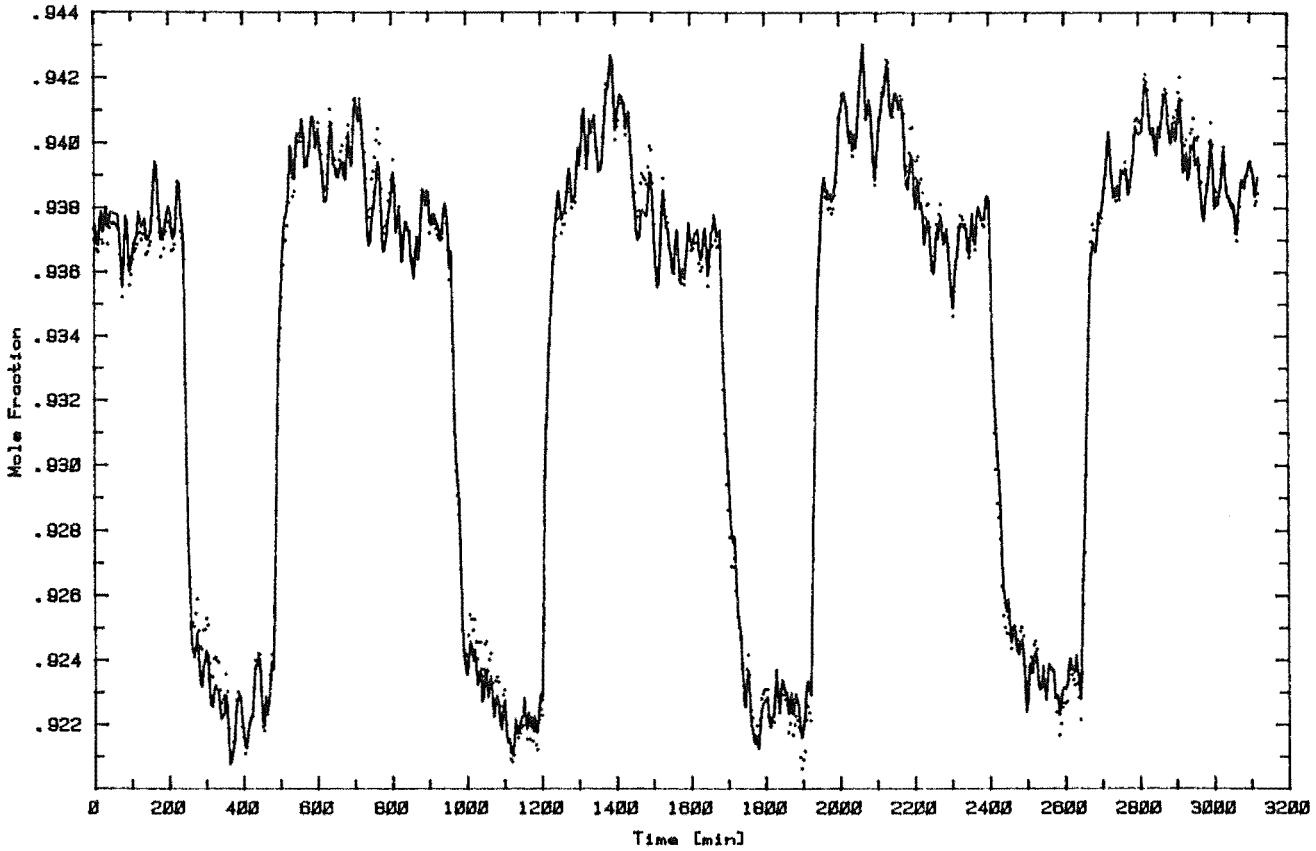
Model Description	Tops	Bottoms
4th order linear	$9.5*10^{-7}$	$2.2*10^{-5}$
4th order diagonal bilinear	$2.0*10^{-7}$	$3.6*10^{-7}$
4th order bilinear	$2.3*10^{-7}$	$3.6*10^{-7}$

Table 7.11

Graph 7.15a 4th Order Diagonal Bilinear Model of Bottoms Composition



Graph 7.15b 4th Order Diagonal Bilinear Model of Tops Composition



Key: ····· Simulated Column Output
—— Identified Model Output

The last series of tests was aimed at finding the main cause of the differences between the simulated column and the bilinear models. For these tests, the simulation was modified from that described in Chapter 3. The initial steady state solution remained unchanged, but for the dynamic solution the equilibrium relation for each plate was described by an equation of the form :

$$y_i^* = k_i x_i + j_i \qquad (7.10)$$

Previously, the liquid/vapour equilibrium had been determined by fitting cubic splines through experimental data points.

Fourth order models of various types were identified using input/output data generated with the modified simulation and the input function shown in Graph 7.1. The variances of the differences between the simulated outputs and the identified model outputs are shown in Table 7.12. These show a considerable reduction when compared with those of the original simulation in Table 7.3. The simulated and the identified model outputs for a fourth order diagonal bilinear model are shown in Graph 7.16. The simulated and the predicted outputs for a fourth order diagonal bilinear model corresponding to the input functions shown in Graph 7.11 are shown in Graph 7.17. The ability of this model to predict the output of the modified simulation compares favourably with that for the original simulation in Graph 7.13.

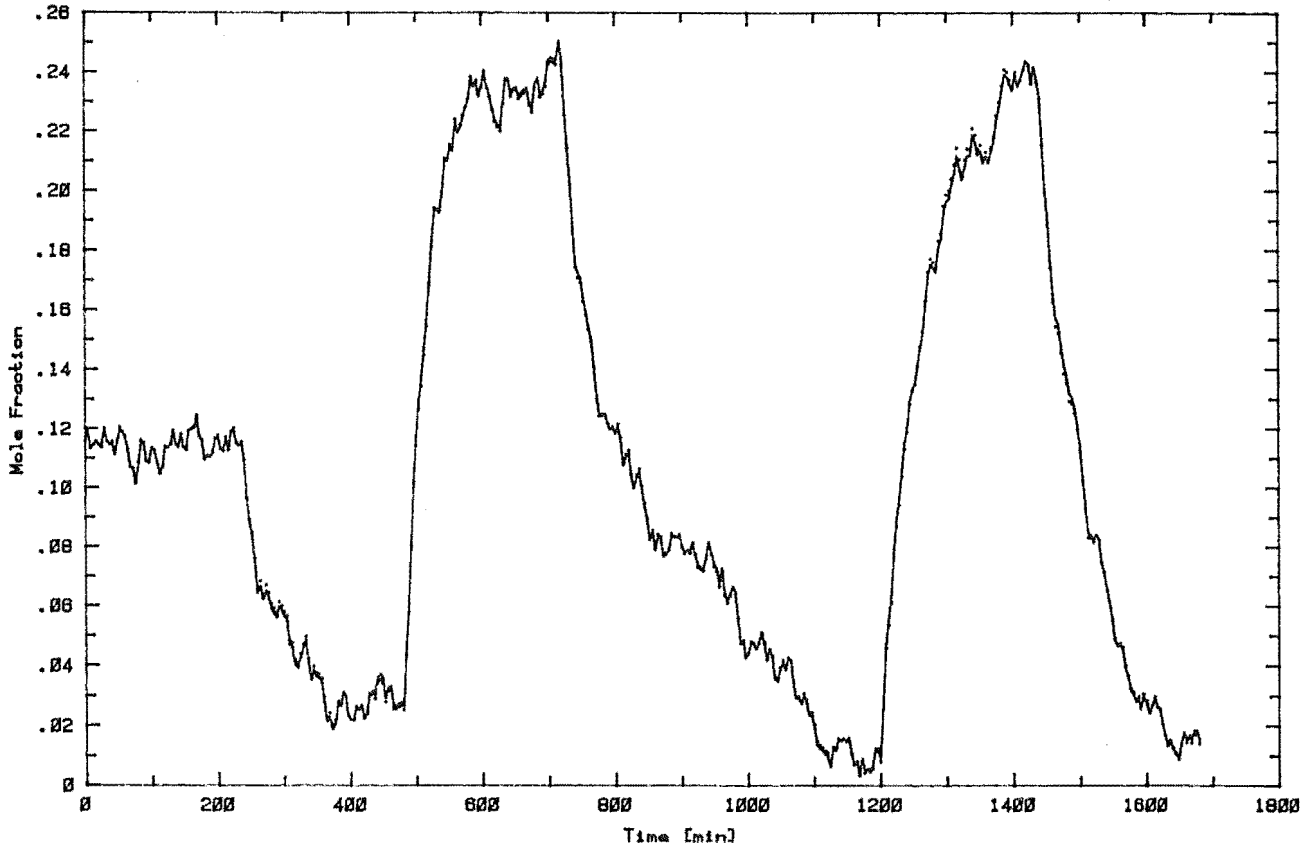
Variances of Model Errors

for Modified Simulation

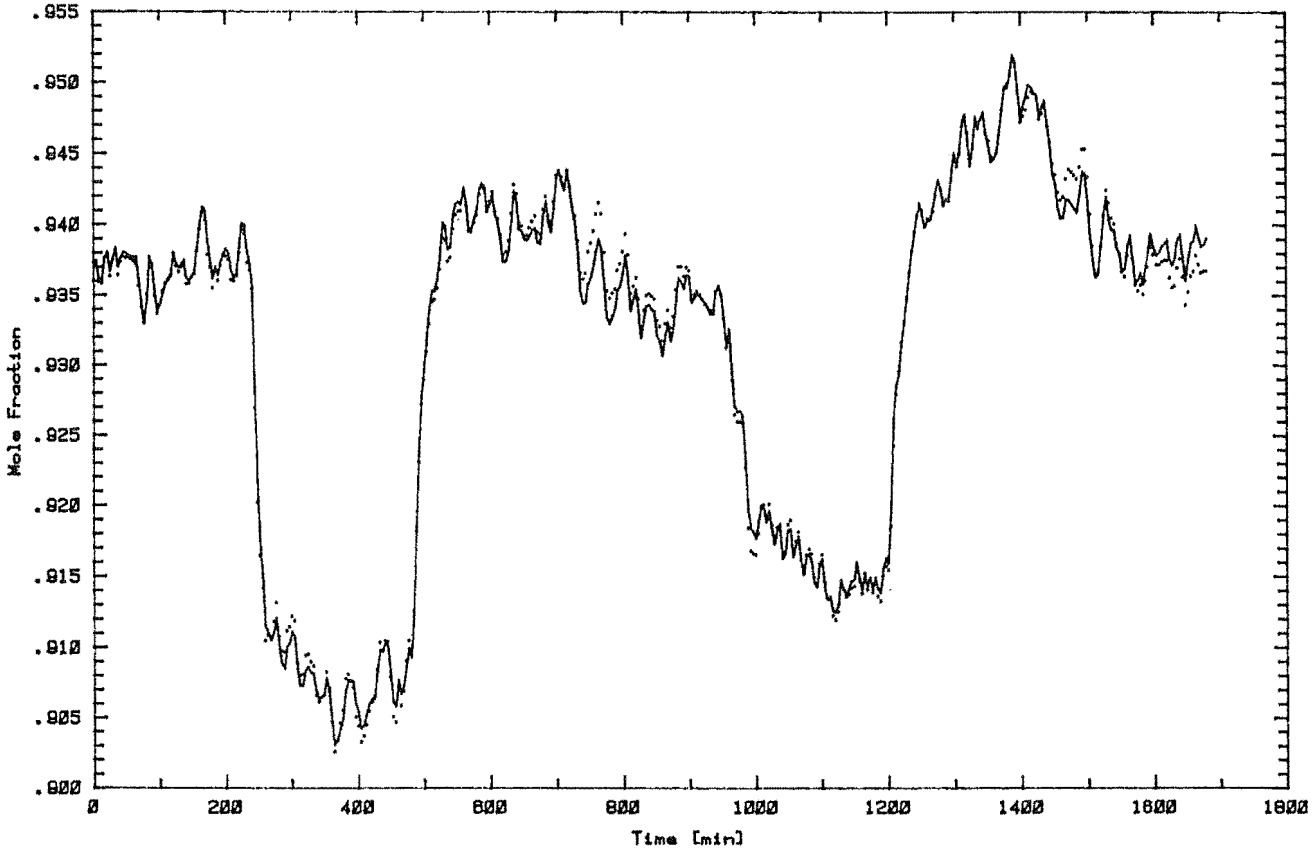
Model Description	Tops	Bottoms
4th order linear	$2.3*10^{-6}$	$1.1*10^{-4}$
4th order diagonal bilinear	$7.0*10^{-7}$	$1.2*10^{-6}$
4th order bilinear	$6.8*10^{-7}$	$9.7*10^{-7}$

Table 7.12

Graph 7.16a 4th Order Diagonal Bilinear Model of Bottome Compoeition

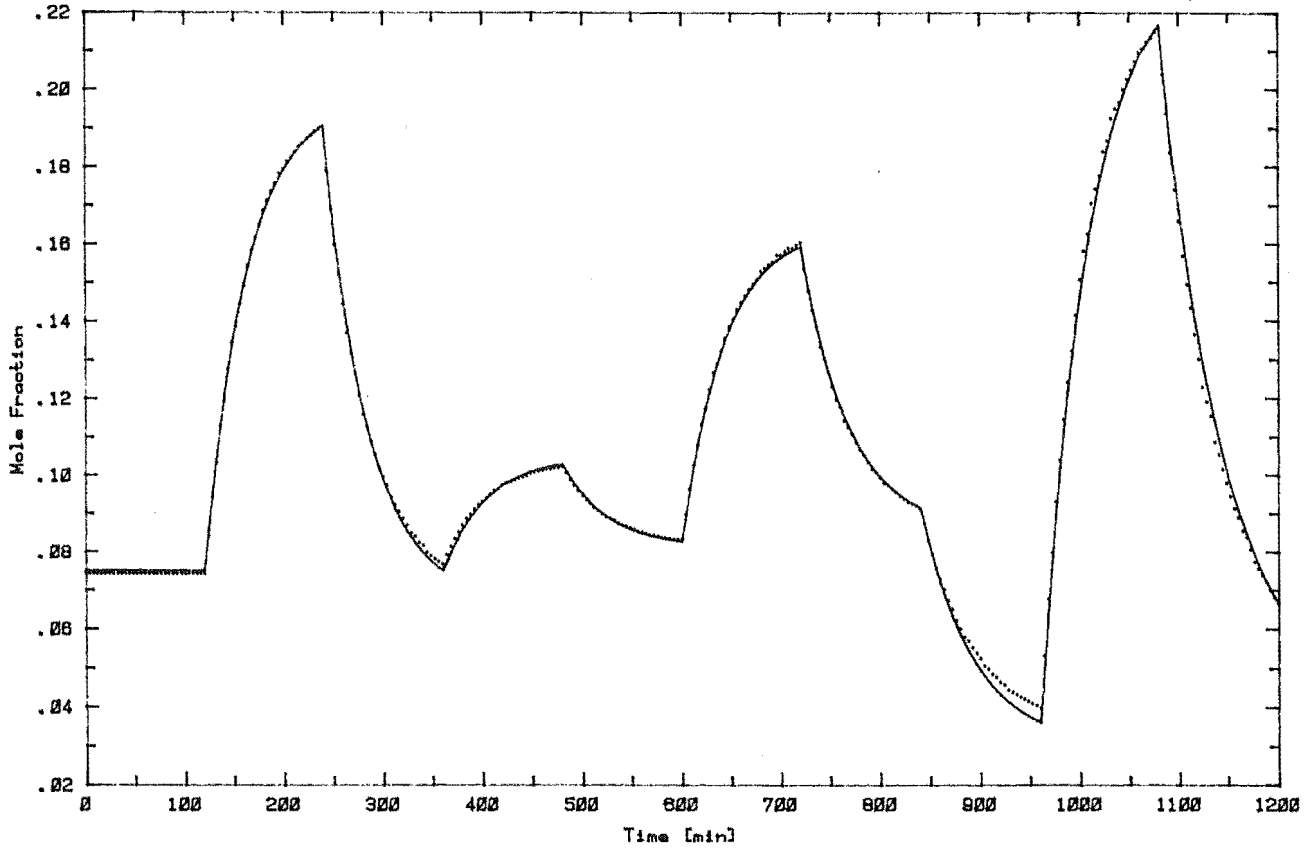


Graph 7.16b 4th Order Diagonal Bilinear Model of Tops Composition

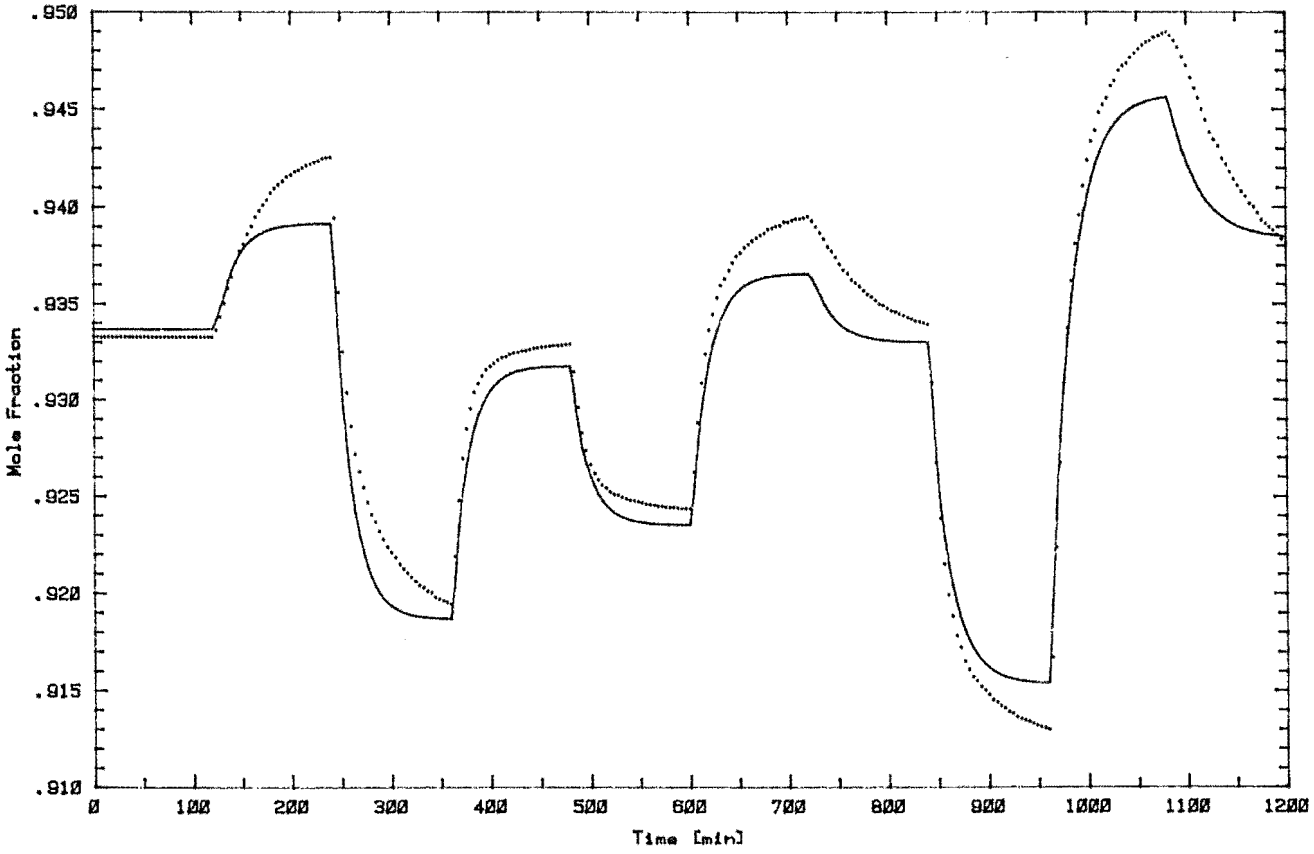


Key: ····· Modified Simulated Column Output
—— Identified Model Output

Graph 7.17a 4th Order Diagonal Bilinear Model of Predicted Bottoms Composition



Graph 7.17b 4th Order Diagonal Bilinear Model of Predicted Tops Composition



Key: ····· Modified Simulated Column Output
—— Identified Model Output

7.4 DISCUSSION

The U-D algorithm converged in all the identification runs attempted using simulated data. Its resistance to modelling error was demonstrated by its ability to successfully identify very simple models, such as the second order linear models in this work. It also proved to be robust enough to deal with large complex models, such as the fourth order bilinear models used.

The choice of the best model type and order was expected to be a compromise between the quality of fit and the model complexity. The diagonal bilinear model was the best model type for applications where the column was operated over a range of conditions. It fitted the data as well as the full bilinear model type but with less complexity. This last point is of considerable note for large systems, such as distillation columns, because the number of parameters in diagonal bilinear models only increase proportionally with the model order, and not with the square of the model order.

The quality of the fit of the full bilinear model was slightly poorer in some cases than the diagonal bilinear model. This could have been caused by a number of factors. Firstly, the parameter identification used a recursive estimation technique operating on a limited set of input/output data. These techniques have a finite convergence rate which tends to get slower when more parameters are estimated. Alternatively, the large number of parameters required for high order bilinear models may have caused numerical inaccuracies in the identification algorithm to become significant.

Linear models were shown to be inferior to bilinear models in describing distillation column behaviour over a wide operating region. The fit of bilinear models improved with increase in order, as shown in Table 7.3, but the improvement for linear

models was much less significant. This demonstrated that increasing the order of the linear models could not compensate for their inherent modelling error. As shown in Table 7.4, the gains of the linear models were an average of the values of the simulated column gains over the operating region used for the identification. Hence, these models were really only valid for a small area at the centre of this operating region.

The table of simulated column gains, Table 7.2, shows the general trend that the column gains increase as the column flowrates decrease. This is typical behaviour for a compartmental bilinear system. In addition to this trend, the gain of distillate composition as a function of feed flowrate appeared to fluctuate wildly. In particular, it was small in comparison with other column gains for most operating points, but it became significant at low reflux rates. The reason for this was that, under normal conditions, the sub-cooled feed had little effect on the column flows in the enriching section of the column and hence on the position of the operating line. However, at low reflux rates, the feed was no longer entering at its optimum position and a pinch point developed at the centre of the column. The result of this was to increase the sensitivity of the position of the enriching section operating line to changes in feed flowrate.

The bilinear models were in general more successful at describing the behaviour of the bottoms composition rather than the tops composition. This was especially so if the error variances in Table 7.3 were compared with the amplitude of the transients in the tops and bottoms composition. This was probably due to the fact that the volume of the reboiler was large in comparison with the holdups of the rest of the column. Typically, there were 1500 moles in the reboiler as compared with 500 moles in the rest of the column. Hence, the reboiler tended

to dominate the dynamics of the whole column, essentially acting as a large bilinear tank with the tops composition merely following the changes in the bottoms composition in a manner determined by the equilibrium relations along the column.

The values of the simulated column gains in Table 7.2 showed that there were two factors affecting the changes in parameters; the turn-down of the column and the shifting of the composition profile along the column. A comparison of the values of the gains of the identified bilinear models in Tables 7.5 to 7.10 with the simulated column gains in Table 7.2, showed that the models appeared to be good at following changes due to the turn-down, but not the large fluctuations due to changes in the composition profile. Another experiment with a different set of input functions, as described in Appendix F.1, was used to verify this observation. These new inputs had the same turn-down ratio as those used previously, but the range of the changes in composition profile down the column was halved. In terms of an operating region being a truncated cone, the height of the cone was left the same but the size of the base reduced. The large reduction in the variances of the model errors for the new operating region, as shown in Table 7.11, confirmed that the second operating region more closely reflected the valid operating region for the bilinear models.

When the form of the equilibrium relation used in the dynamic simulation was changed from one based on experimental points to one based on the assumptions used to derive the bilinear model, the variances of the model errors were significantly reduced. This can be seen by comparing Table 7.12 with Table 7.3. This implied that a major limitation of bilinear models of distillation columns was the overly simple equilibrium relation used in their derivation. This was particularly so for

the methanol/water system which has a very high relative volatility.

The attempts to predict future column behaviour shown in Graphs 7.12 to 7.14 implied that a good fit of a model to one set of input/output data did not necessarily result in as good a fit for another set of data, even if it was in the same general operating region. This outlined the need to have plant operating data for the whole region over which a good fit was required. It also pointed to a potential danger with more complex models, in that the parameters may be estimated to give a good fit for only one particular set of input/output data. This suggests that non-linearities should be chosen that are suggested by the plant structure rather than using more general non-linear equations.

CHAPTER 8

BILINEAR IDENTIFICATION OF AN EXPERIMENTAL DISTILLATION COLUMN

8.1 INTRODUCTION

In the previous chapter, a bilinear model of a binary distillation column was derived. The parameters of several different orders and types of model were identified for simulated data, and the validity region for bilinear models of distillation columns was studied.

In this chapter, data generated using the experimental distillation column described in Chapter 2, was used. The aim was to test if the identification technique was capable of dealing with uncertain dead time, process noise and modelling error. For the experimental column used in this work, dead time resulted from flows inside the column and inside the product lines. In addition, the use of low order models lumped the high order effects with the dead time. For other experimental columns, it may also result from delays in composition sensors such as chromatographs and refractometers.

The experimental column was operated under open loop conditions, with the VAX minicomputer supplying the set points for the feed, reflux, and steam condensate flowrates. The resultant column outputs were then read and saved in a data file on the VAX, ready for subsequent identification. The feed composition was maintained at 0.5 mole fraction by switching the solenoid valves between the product tanks and the feed tanks. The sampling rate was set at 4.0 minutes, in accordance with the guidelines in

Section 4.10. Simple step responses showed that at this sampling rate, the various dead times were less than one sampling interval.

8.2 IDENTIFICATION

The U-D algorithm was used to identify the parameters in the various models in a similar manner to that used for the simulated data in the previous chapter. As mentioned previously, this was a recursive identification algorithm, but was used batchwise. This meant that it could later be used in real-time to follow changes in the column parameters.

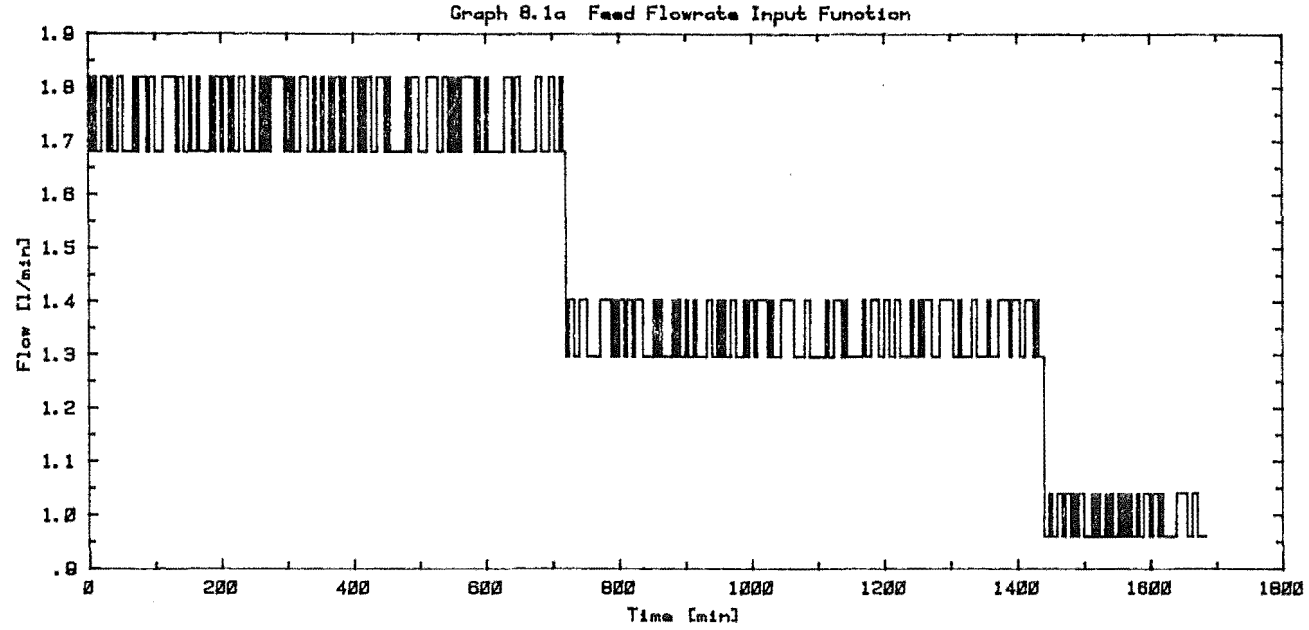
The identified models were verified by several means. The experimental and model outputs for the data used in the identification were compared graphically. The quality of fit for different models was measured by evaluating the variance of the difference between the experimental and model outputs. The ability of the identified models to predict future column behaviour was tested by comparing the experimental and model outputs for an input function different from that used for the identification.

8.3 RESULTS

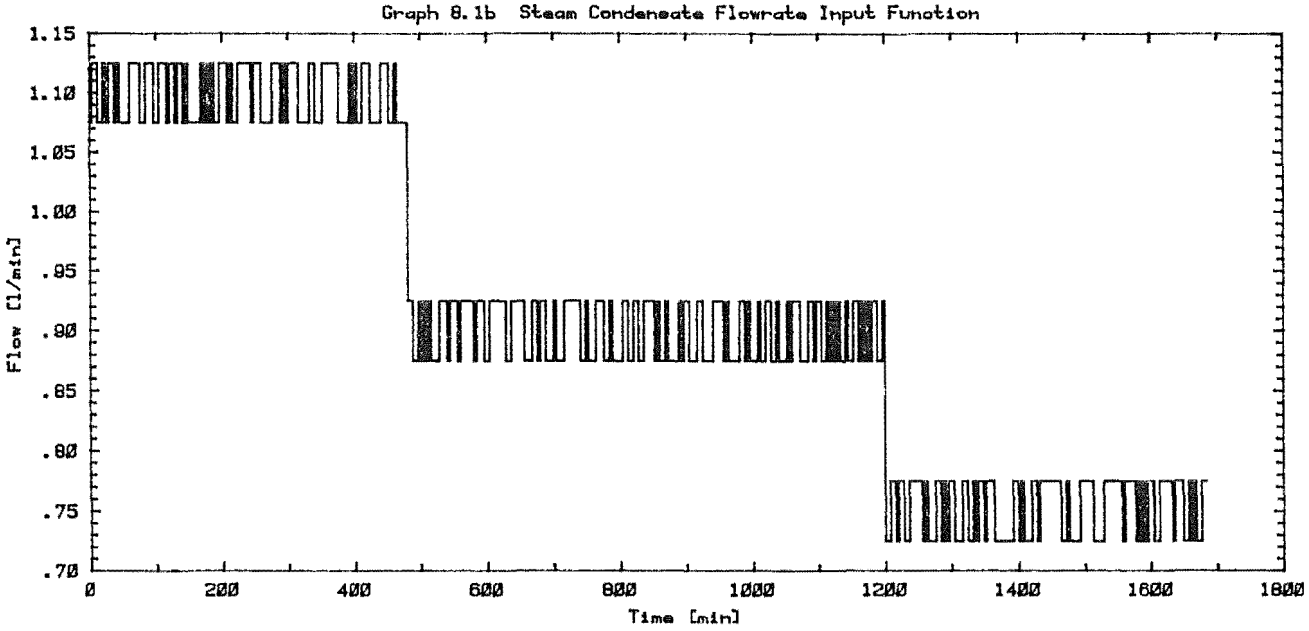
The input functions shown in Graph 8.1 were applied to the experimental distillation column described in Chapter 2. Several models of different types and orders were identified. The variances of the differences between the identified model outputs and the experimental column outputs are shown in Table 8.1. Graphs 8.2 to 8.4 show the outputs of the experimental column and of several identified fourth order models of different types.

A comparison of Graphs 8.2 to 8.4 shows that the bilinear models are better at fitting the experimental data than the linear model. Table 8.1 confirms this, and also shows improvement in fit with increasing model order.

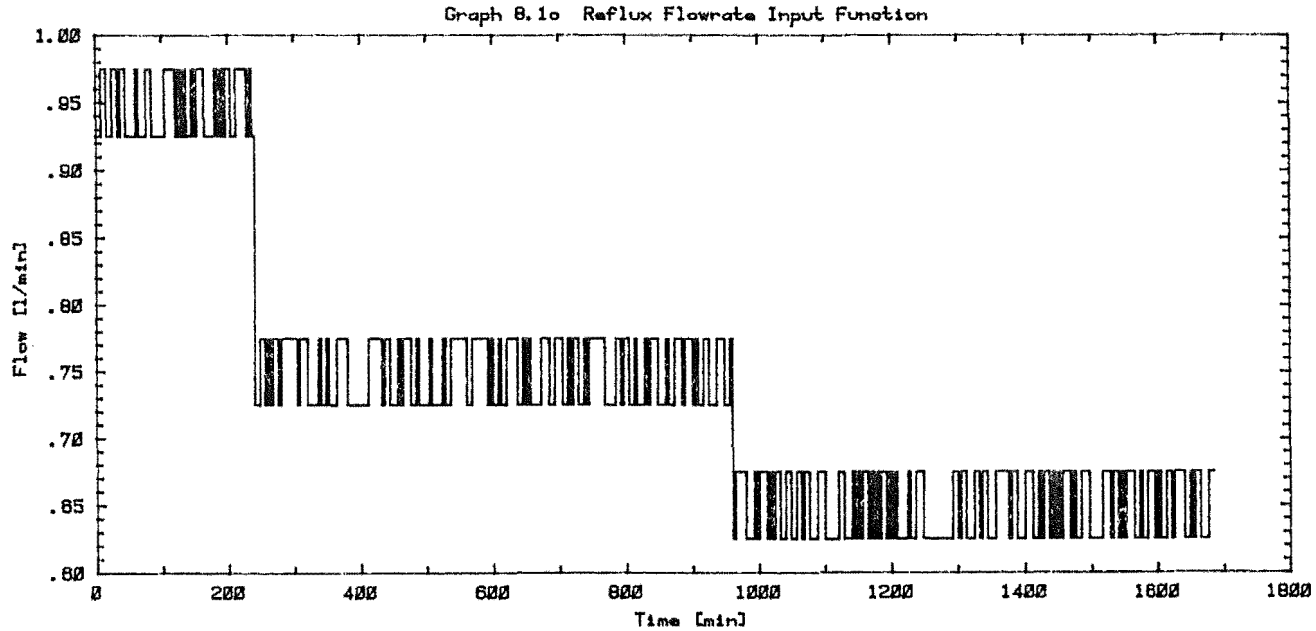
Graph 8.1a Feed Flowrate Input Funotion



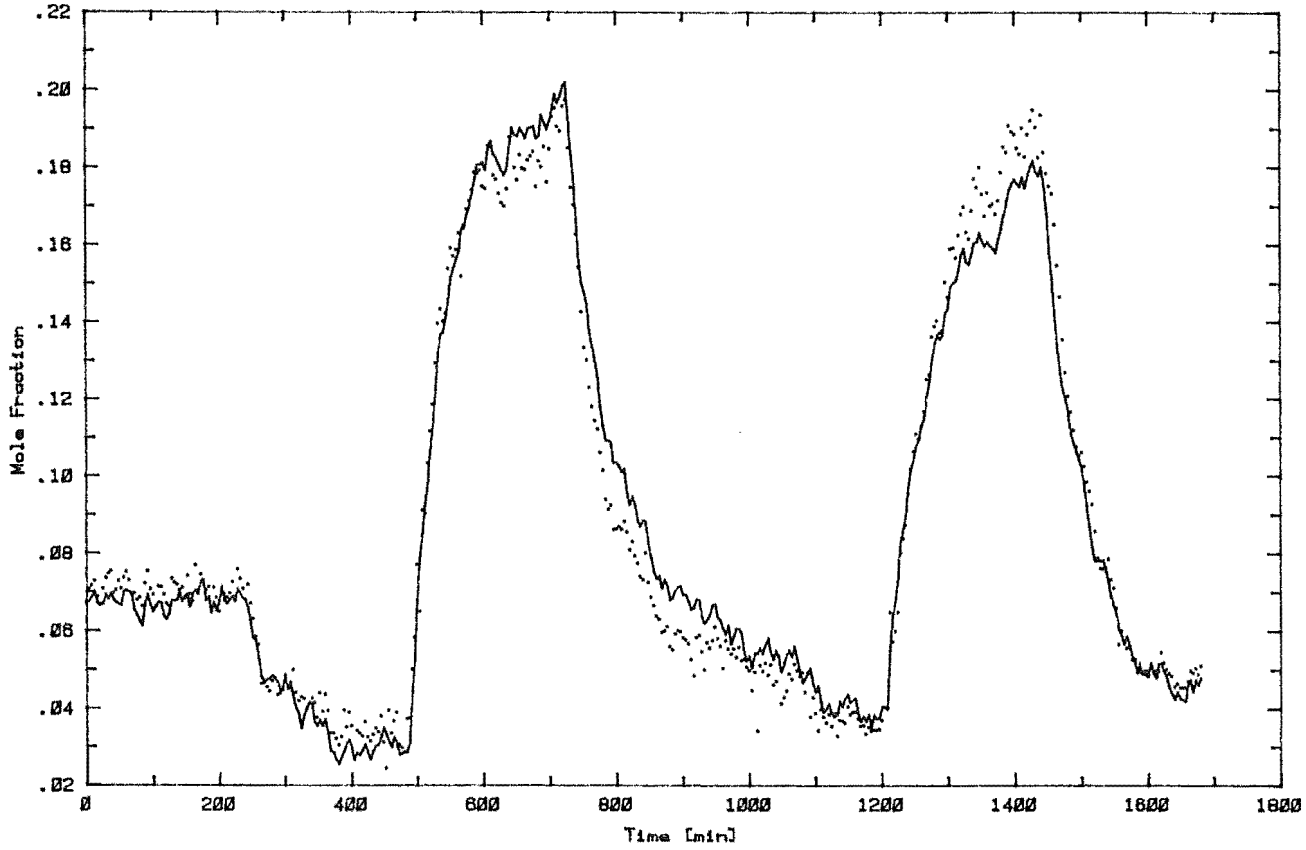
Graph 8.1b Steam Condensate Flowrate Input Funotion



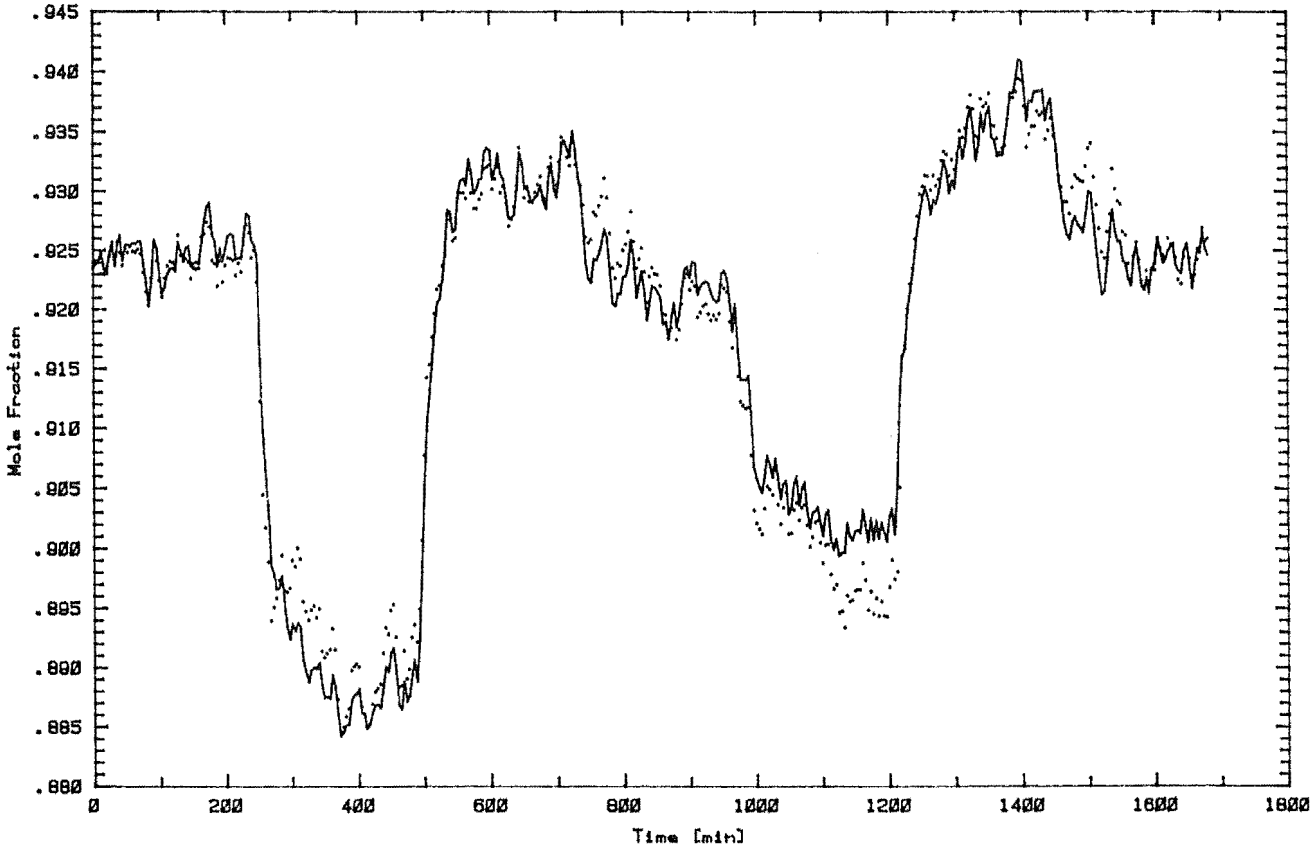
Graph 8.1c Reflux Flowrate Input Funotion



Graph 8.2a 4th Order Linear Model of Bottoms Composition

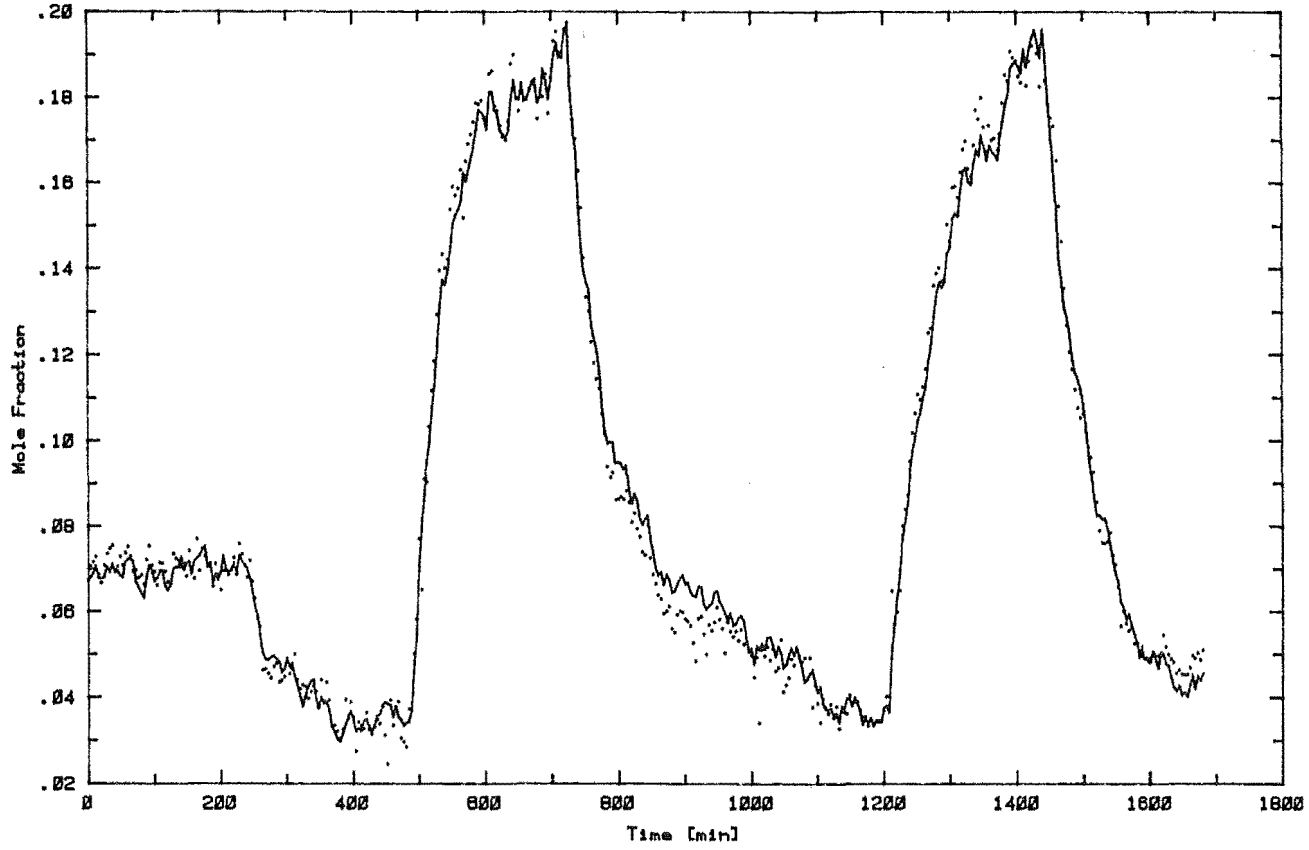


Graph 8.2b 4th Order Linear Model of Tops Composition

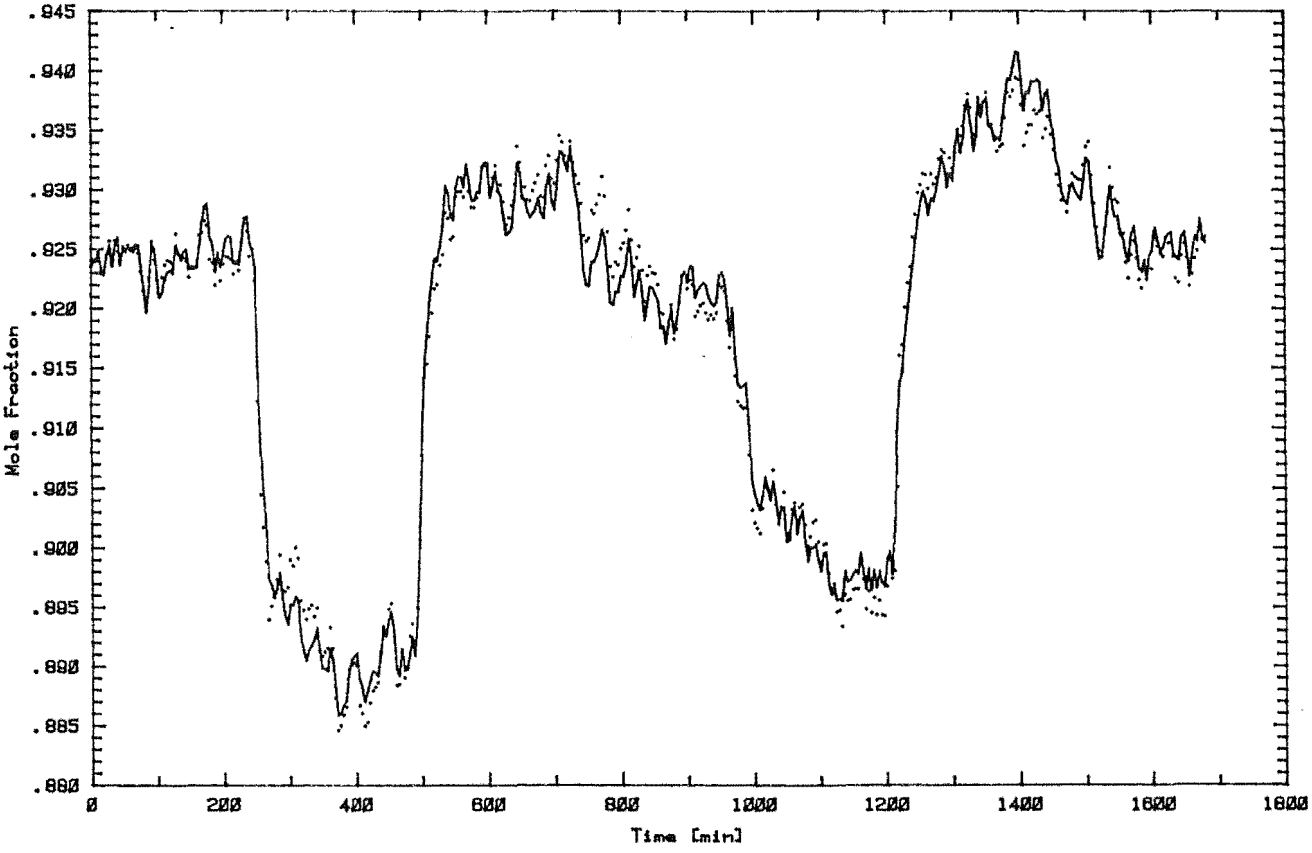


Key: ····· Experimental Column Output
—— Identified Model Output

Graph 8.3a 4th Order Diagonal Bilinear Model of Bottome Composition

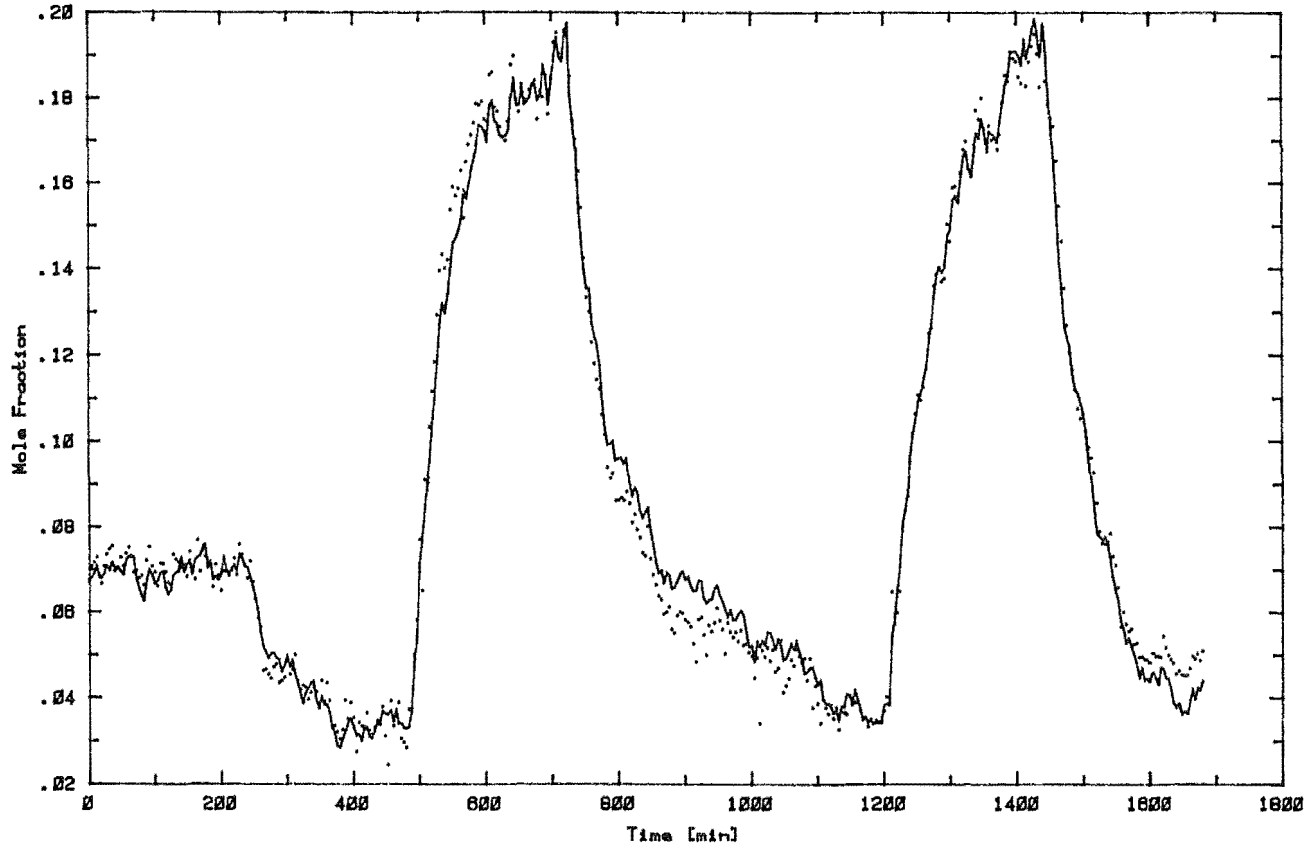


Graph 8.3b 4th Order Diagonal Bilinear Model of Tops Composition

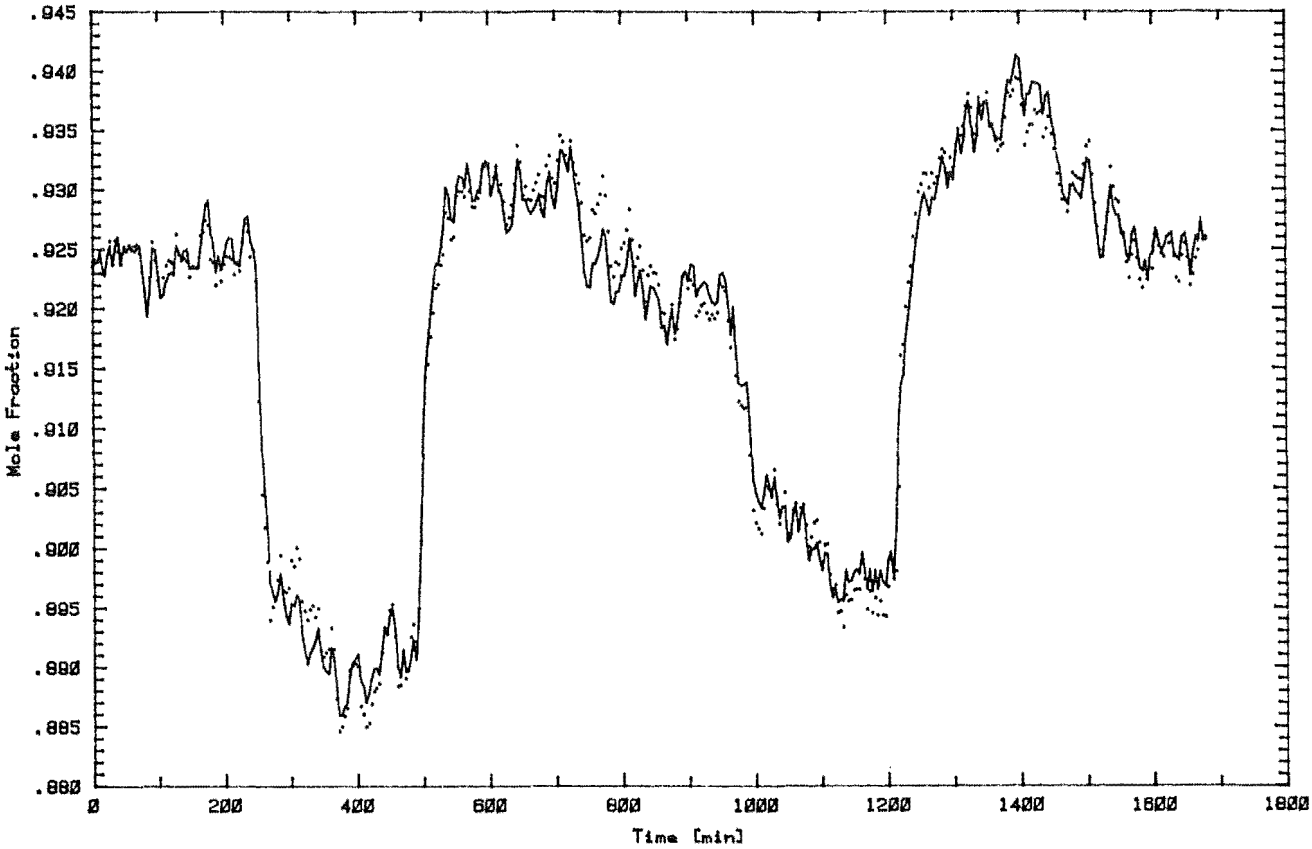


Key: ····· Experimental Column Output
—— Identified Model Output

Graph 8.4a 4th Order Bilinear Model of Bottome Composition

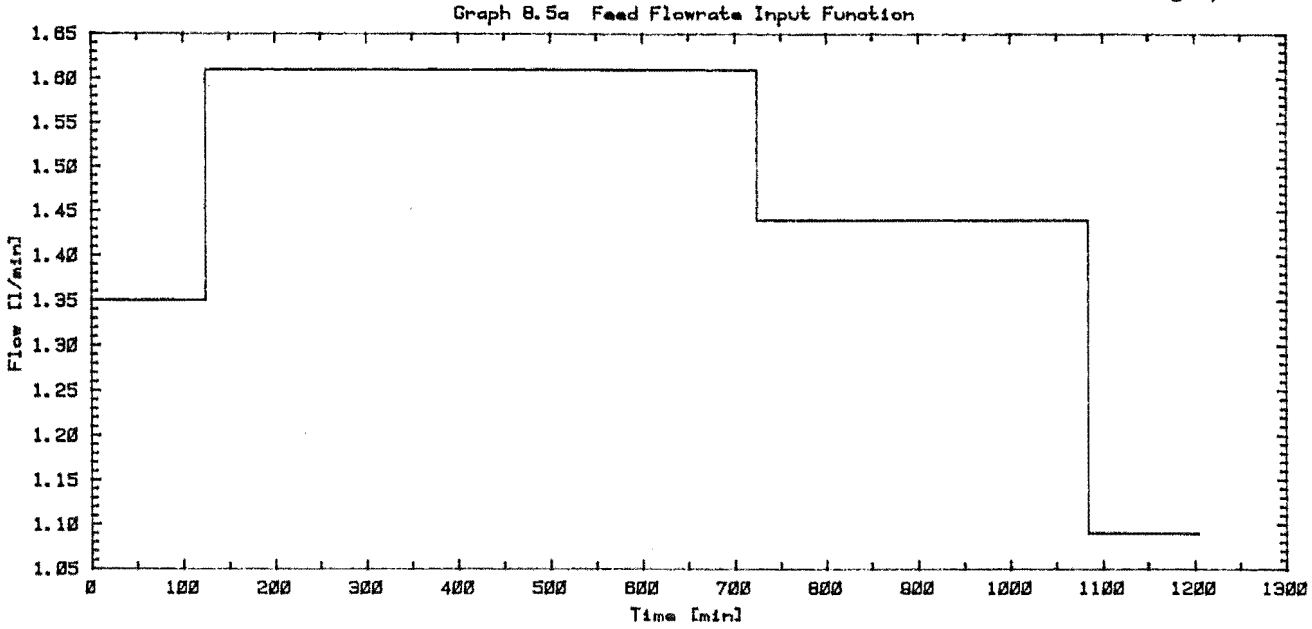


Graph 8.4b 4th Order Bilinear Model of Tops Composition

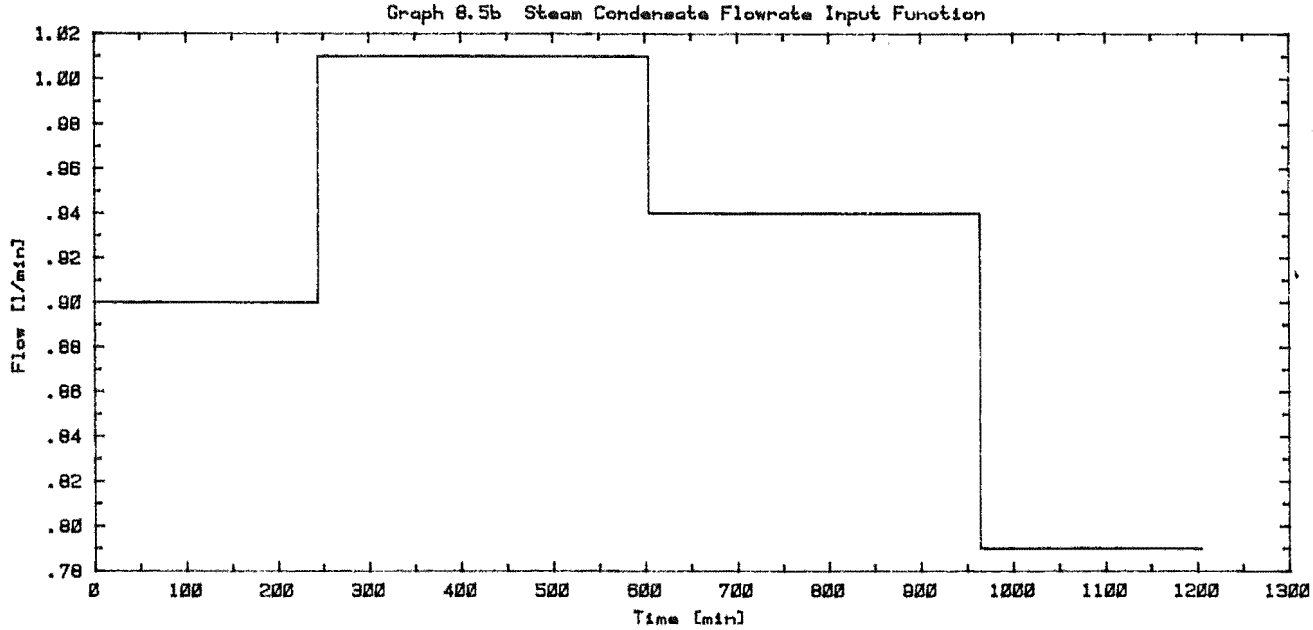


Key: ····· Experimental Column Output
—— Identified Model Output

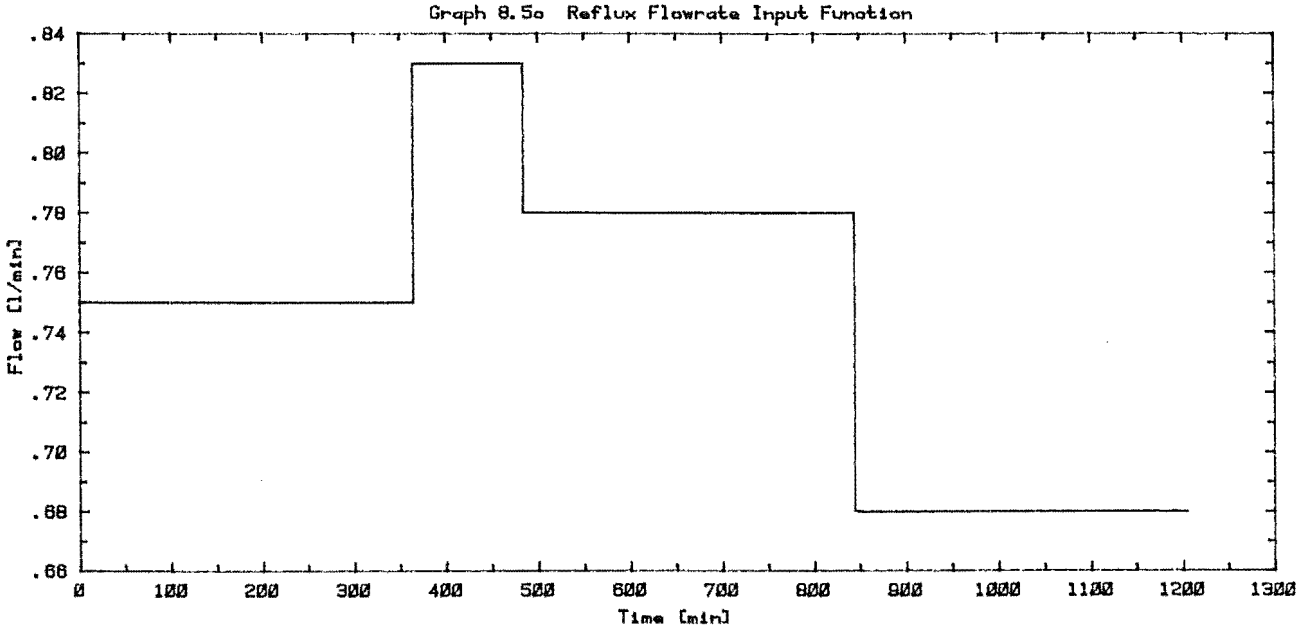
Graph 8.5a Feed Flowrate Input Function



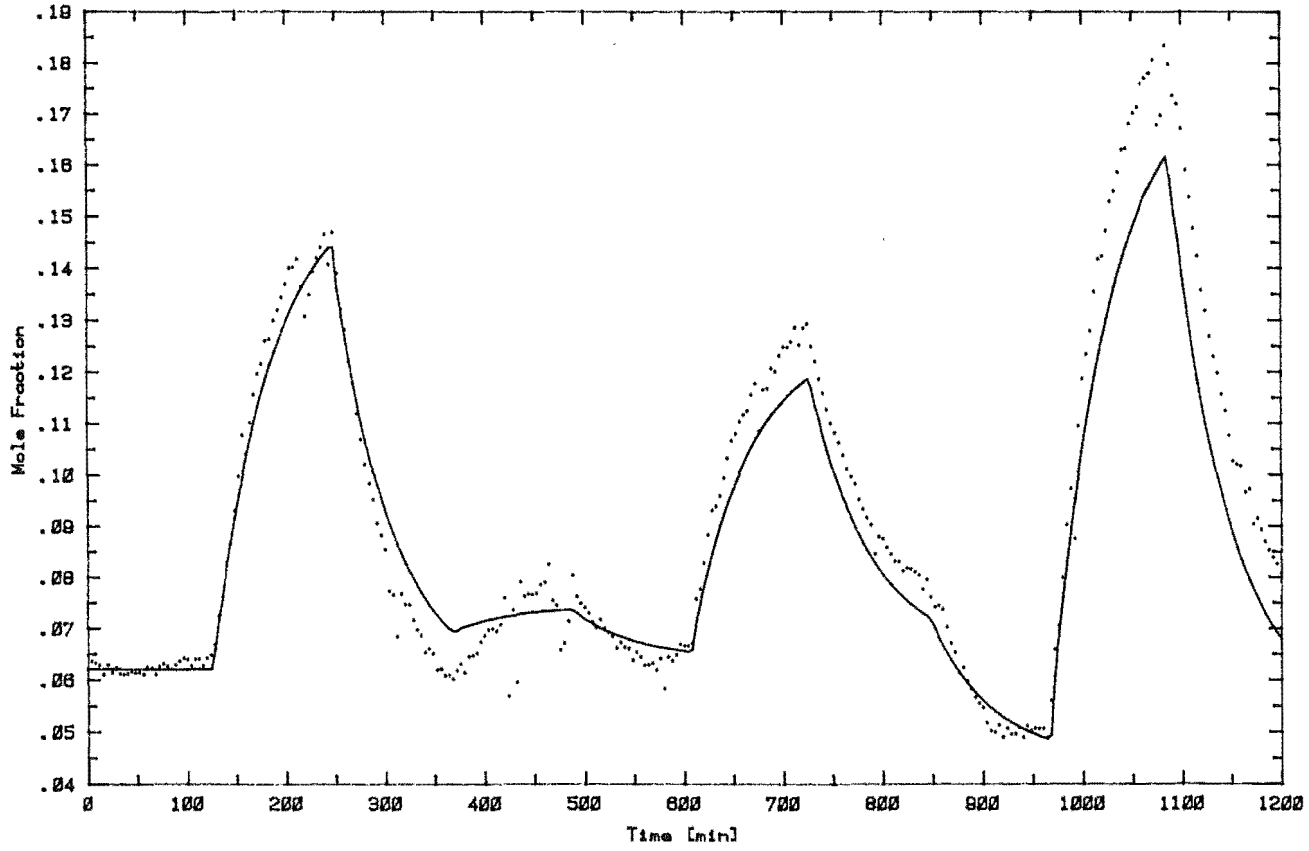
Graph 8.5b Steam Condensate Flowrate Input Function



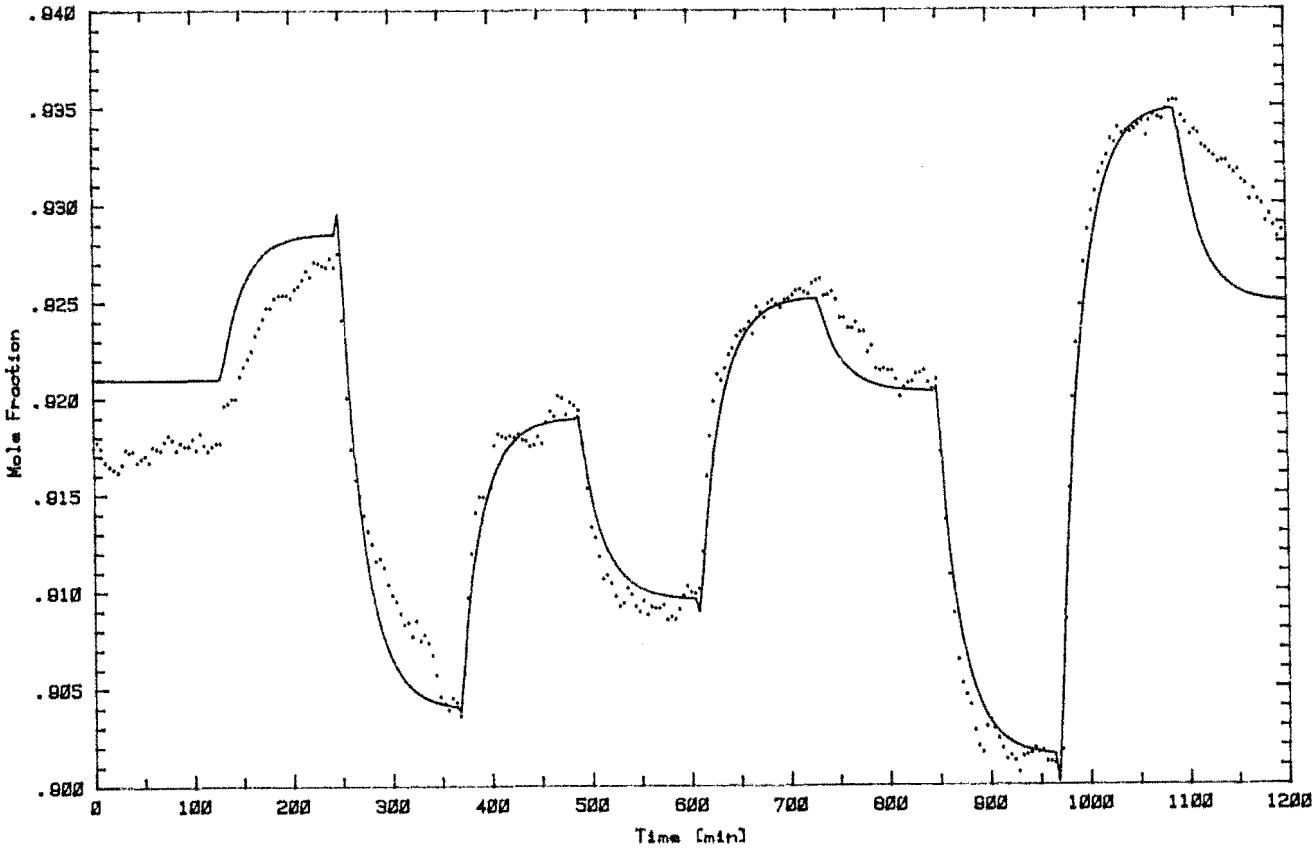
Graph 8.5c Reflux Flowrate Input Function



Graph 8.6a 4th Order Linear Model of Predicted Bottoms Composition

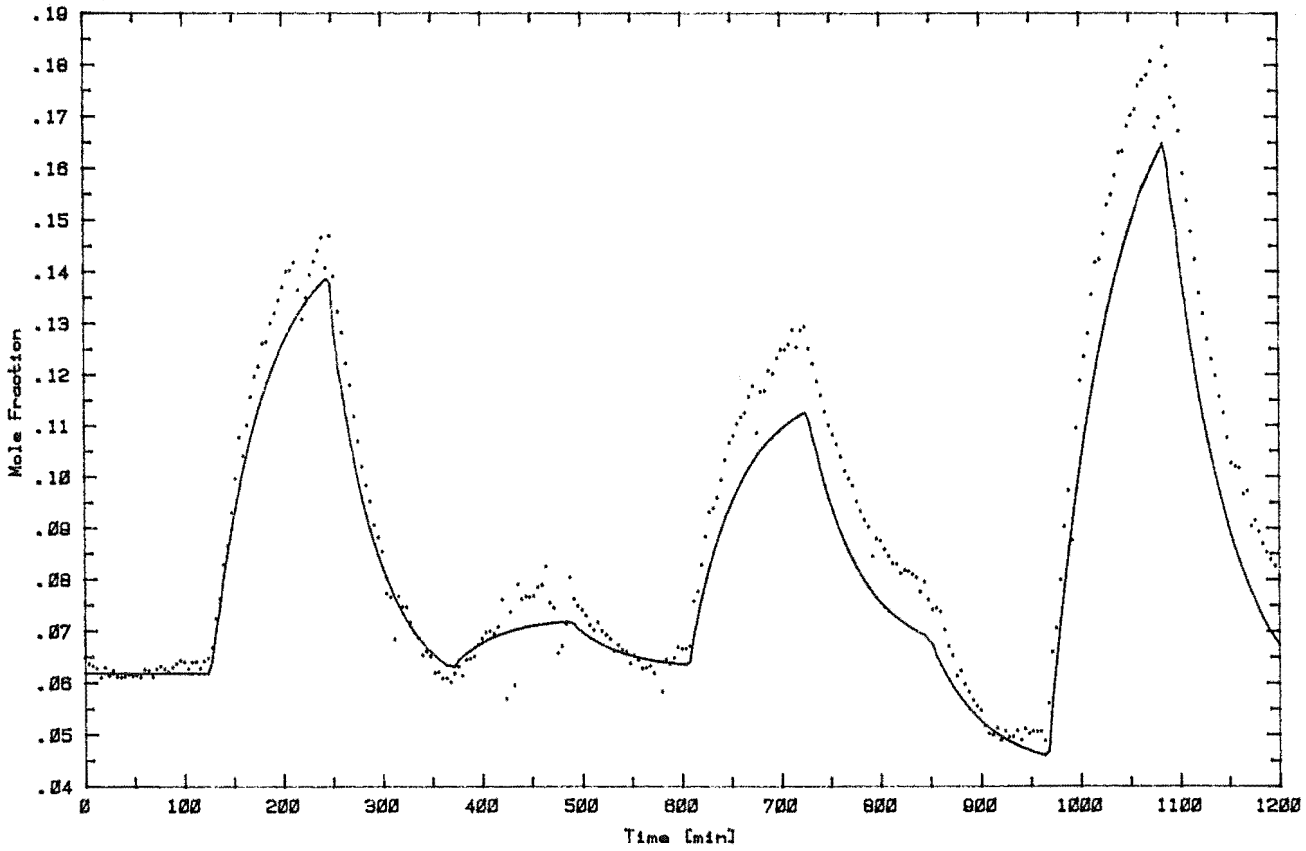


Graph 8.6b 4th Order Linear Model of Predicted Tops Composition

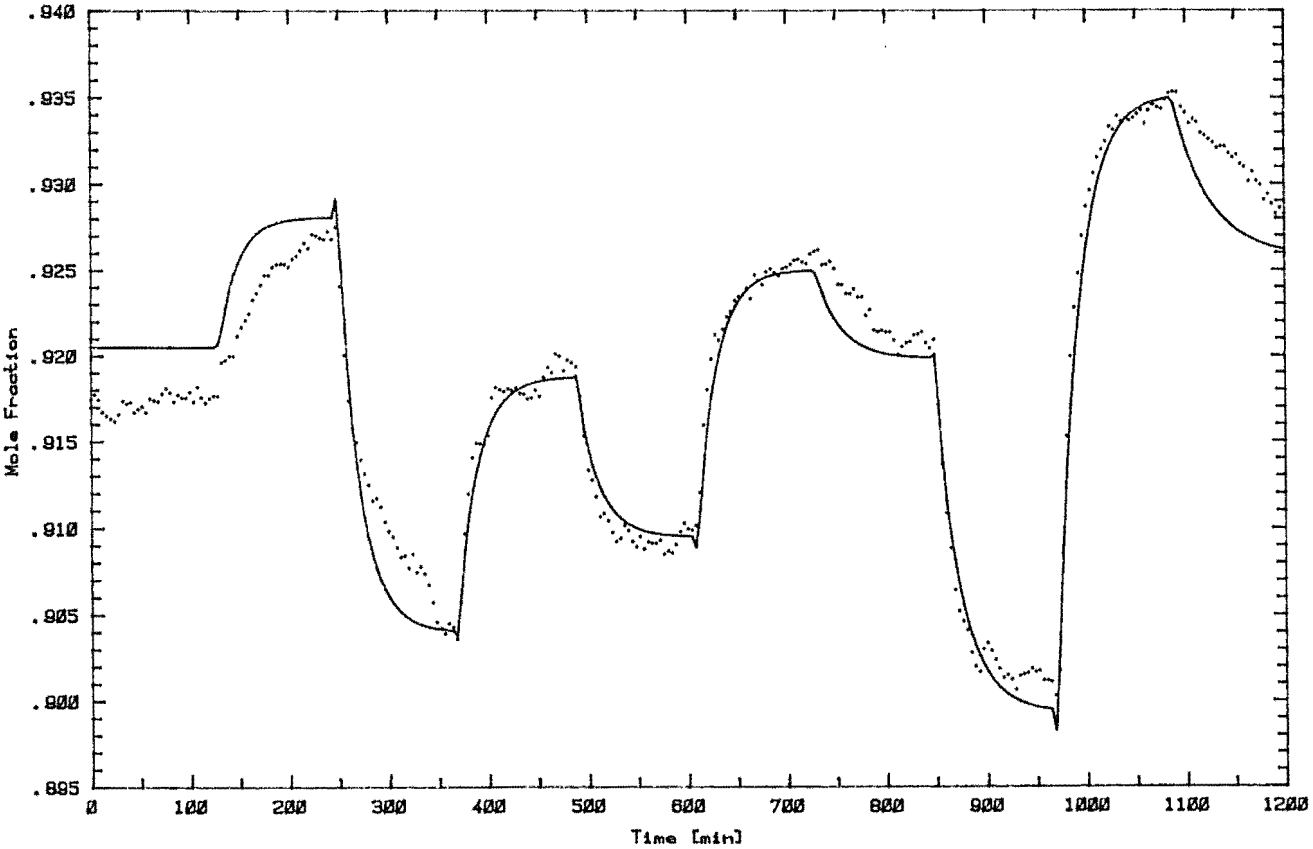


Key: ····· Experimental Column Output
—— Identified Model Output

Graph 8.7a 4th Order Diagonal Bilinear Model of Predicted Bottome Composition

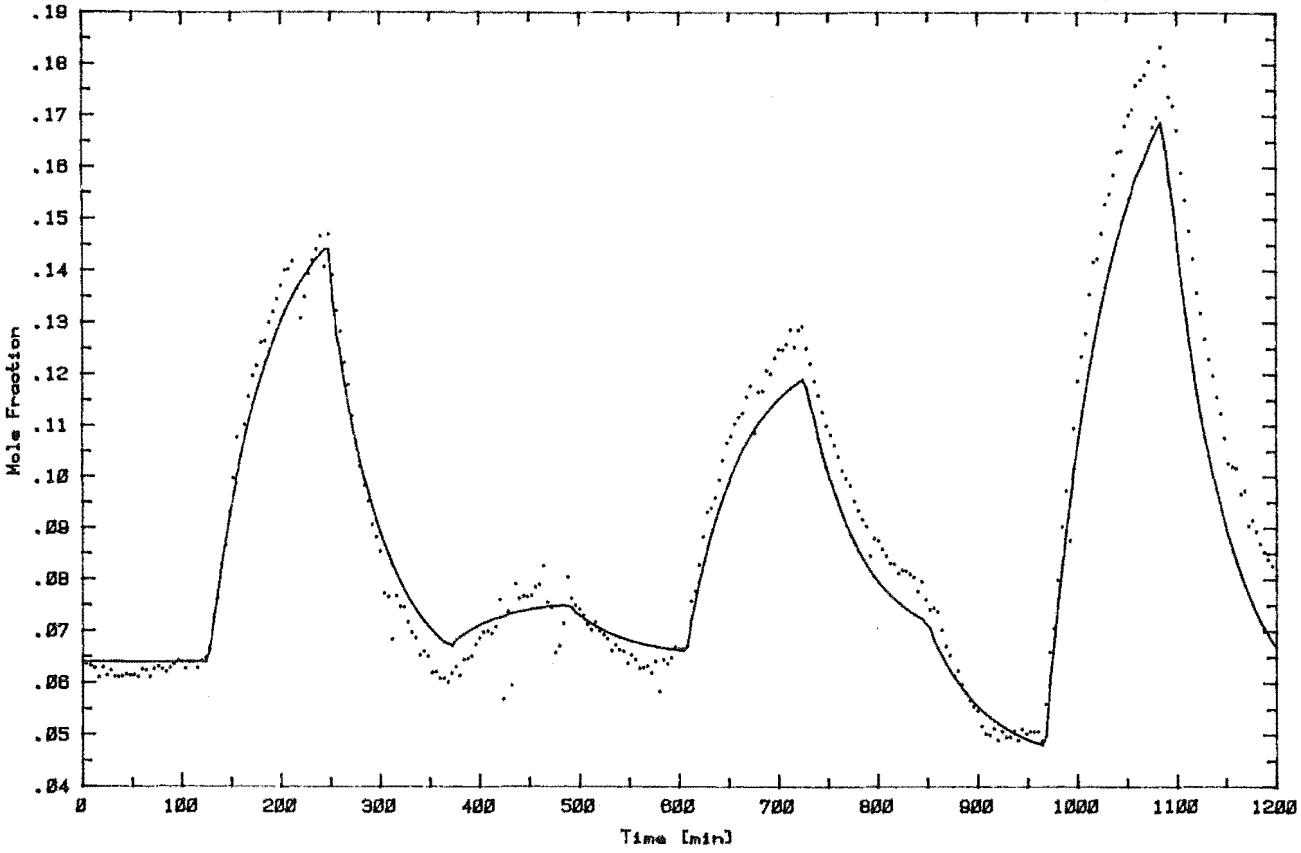


Graph 8.7b 4th Order Diagonal Bilinear Model of Predicted Tops Composition

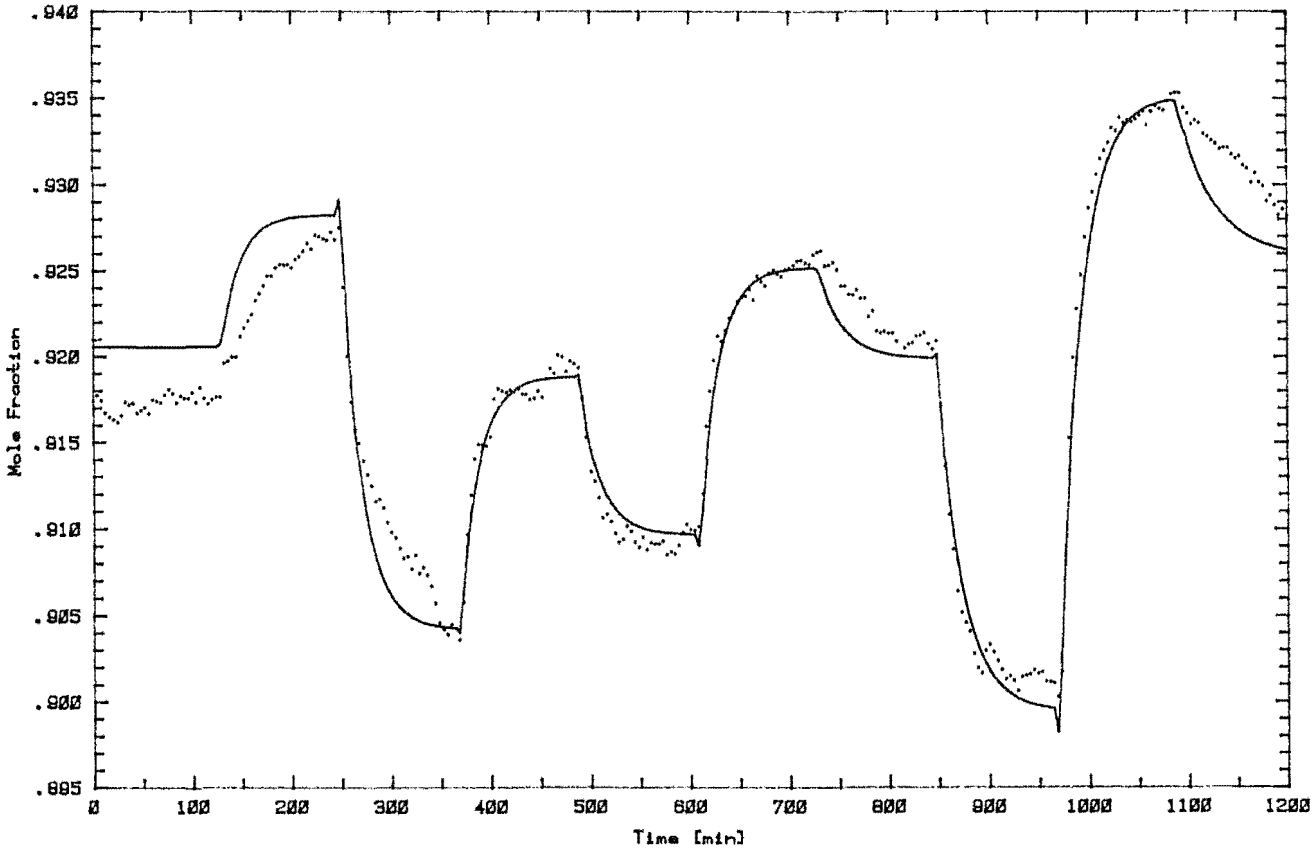


Key: ····· Experimental Column Output
—— Identified Model Output

Graph 8.8a 4th Order Bilinear Model of Predicted Bottoms Composition

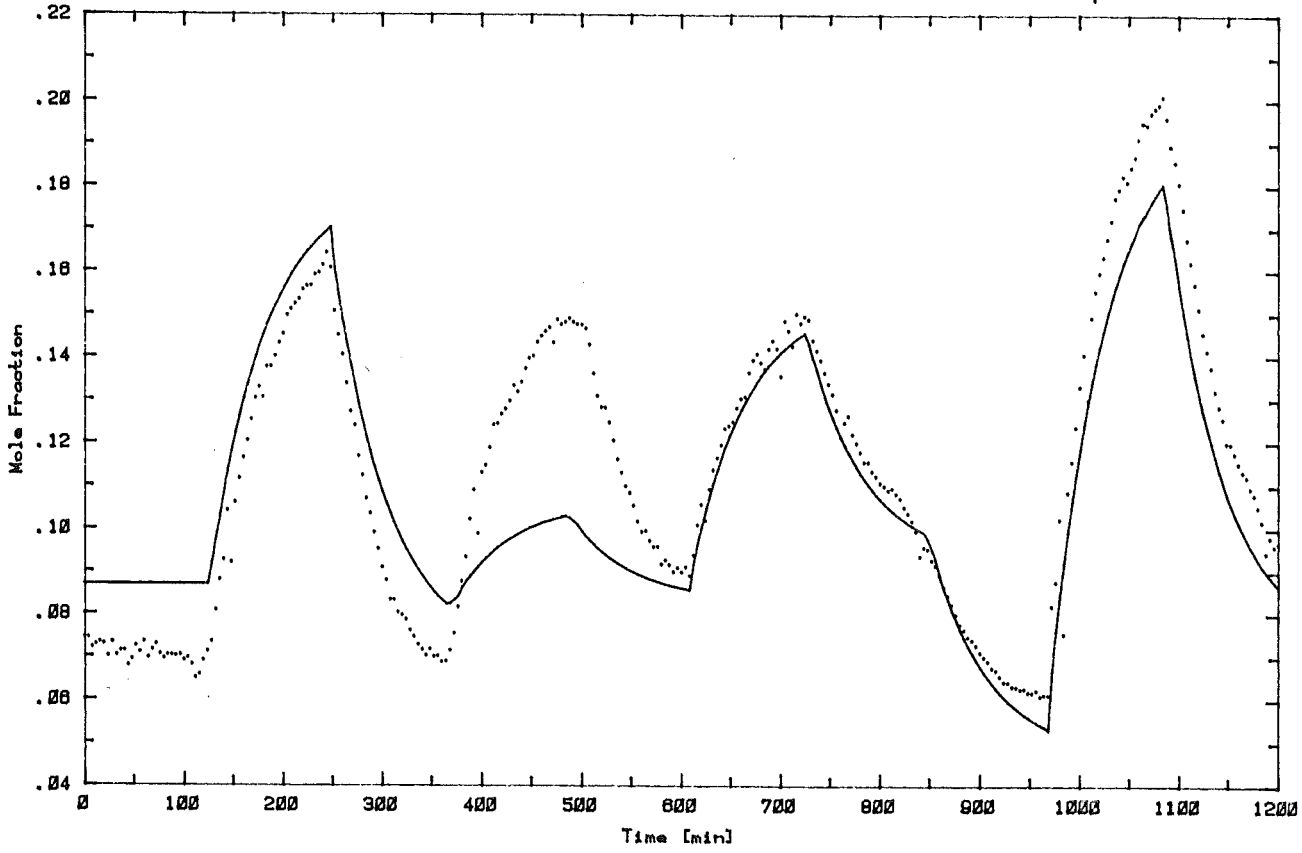


Graph 8.8b 4th Order Bilinear Model of Predicted Tops Composition

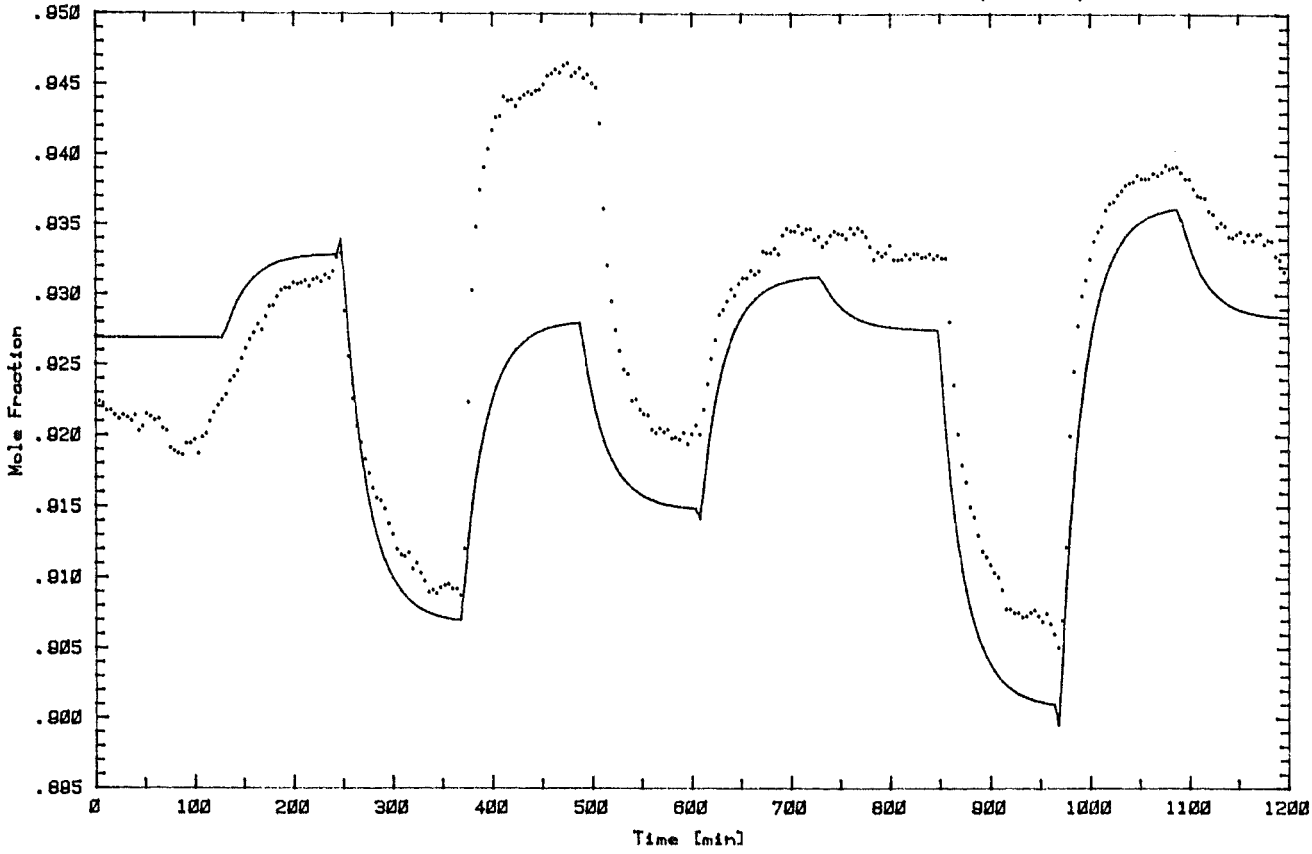


Key: ····· Experimental Column Output
—— Identified Model Output

Graph 8.9a 4th Order Linear Model of Predicted Bottome Composition

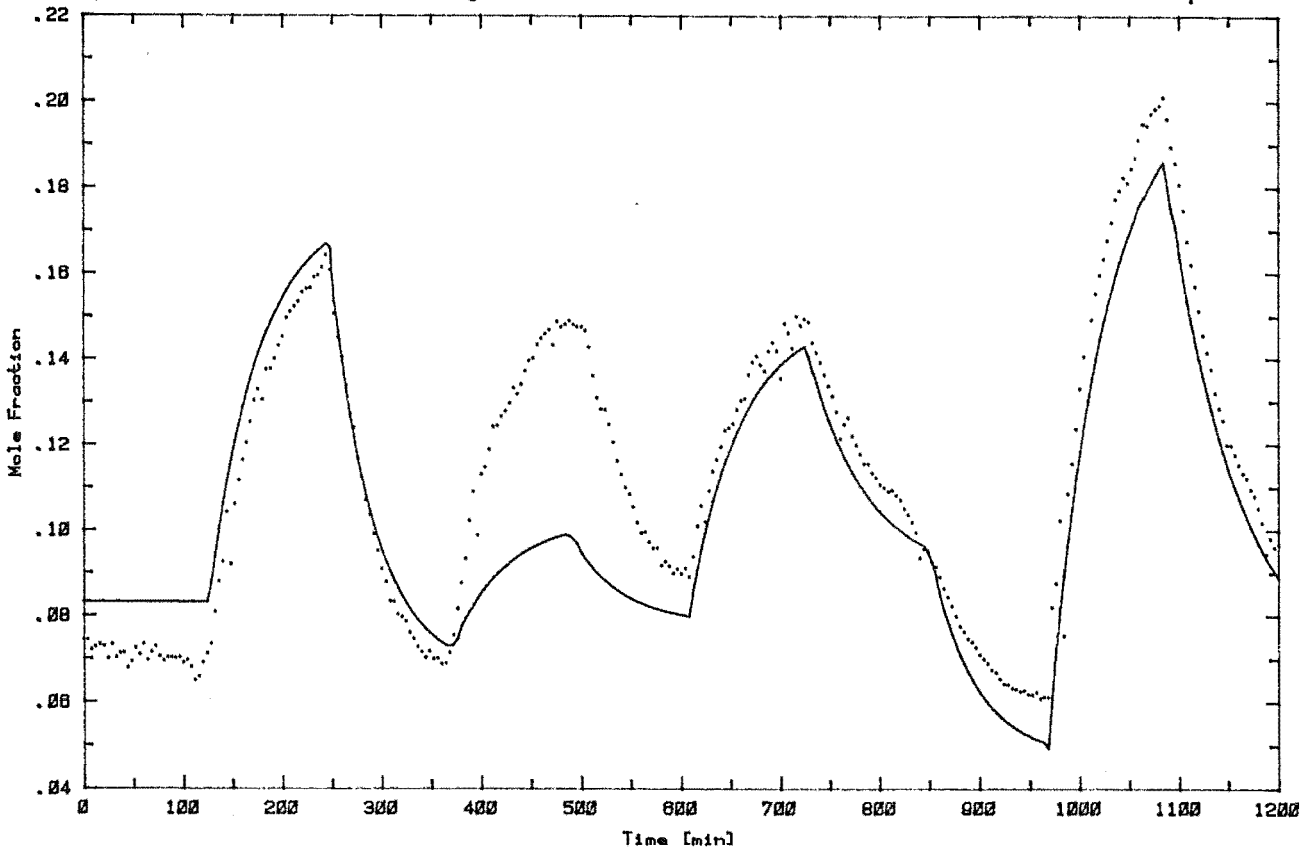


Graph 8.9b 4th Order Linear Model of Predicted Tope Composition

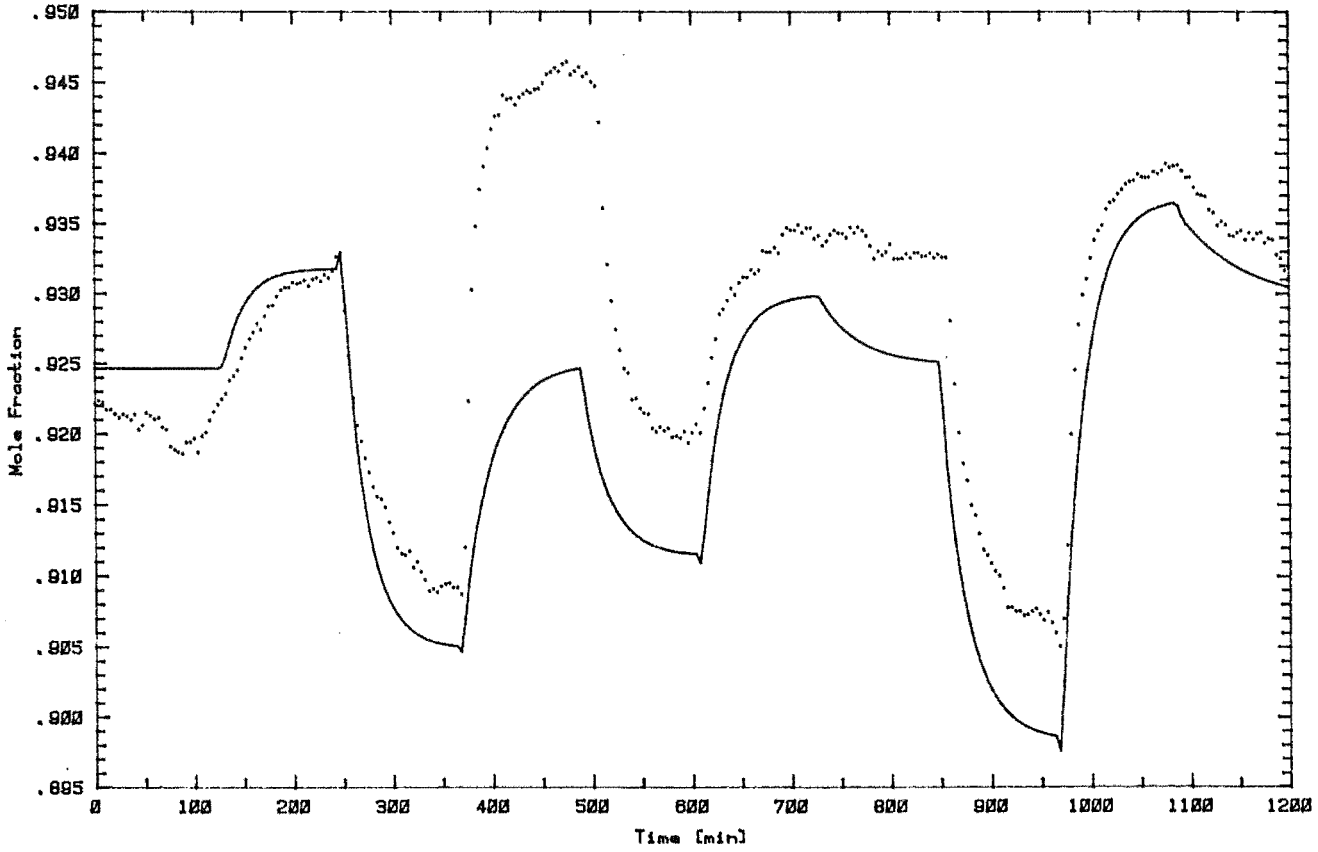


Key: ····· Experimental Column Output
—— Identified Model Output

Graph 8.10a 4th Order Diagonal Bilinear Model of Predicted Bottome Composition

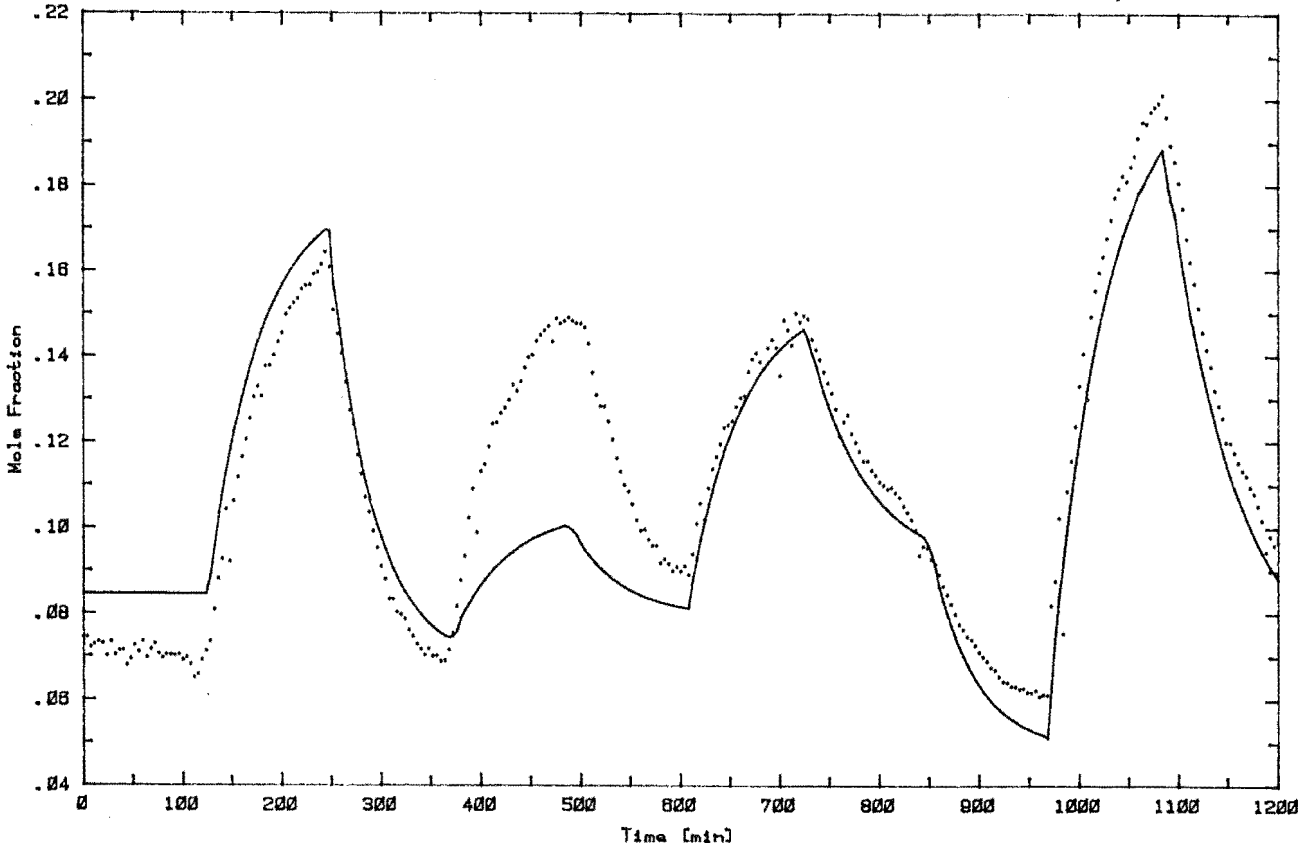


Graph 8.10b 4th Order Diagonal Bilinear Model of Predicted Tops Composition

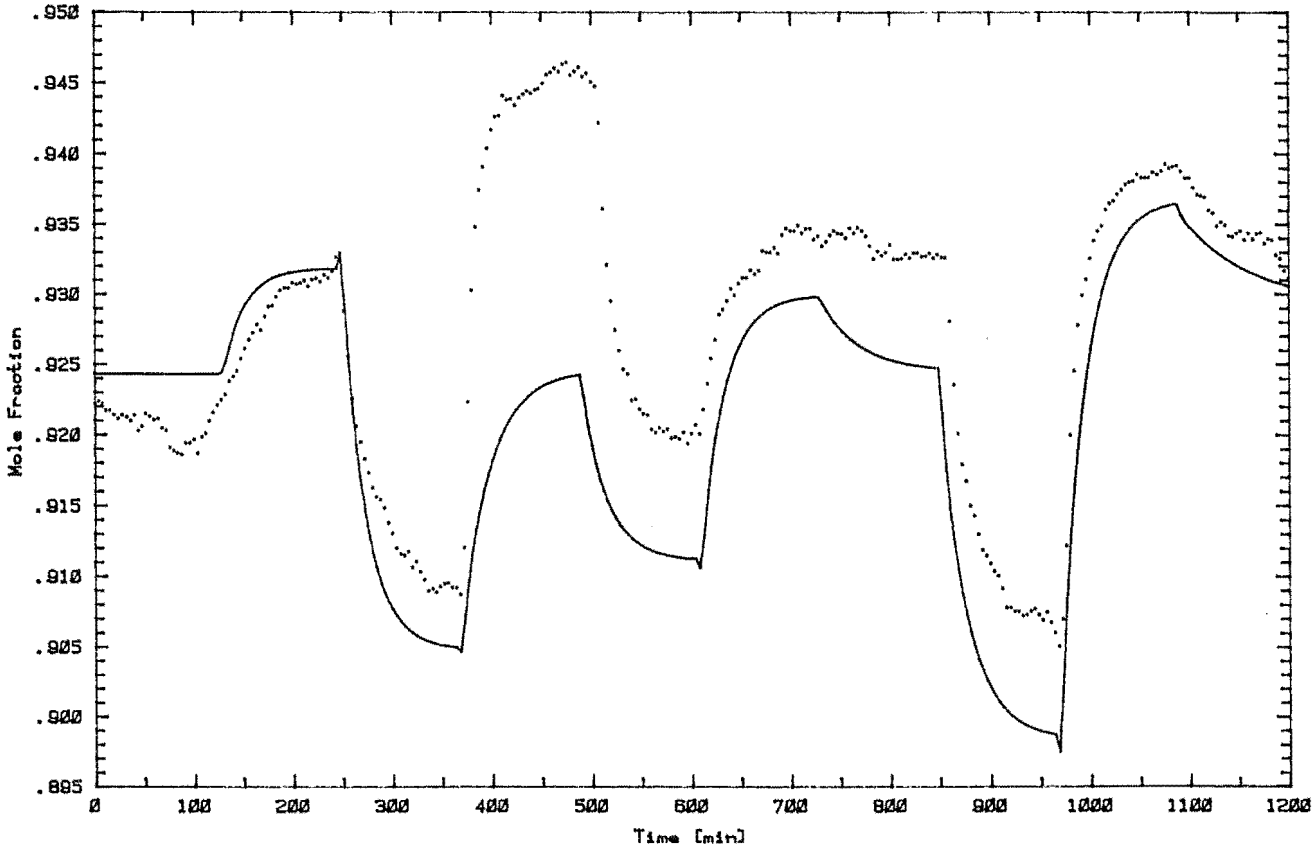


Key: ····· Experimental Column Output
—— Identified Model Output

Graph 8.11a 4th Order Bilinear Model of Predicted Bottoms Composition



Graph 8.11b 4th Order Bilinear Model of Predicted Tops Composition



Key: ····· Experimental Column Output
—— Identified Model Output

Variances of Model Errors

Model Description	Tops	Bottoms
2nd order linear	$6.5*10^{-6}$	$5.8*10^{-5}$
3rd order linear	$6.3*10^{-6}$	$5.7*10^{-5}$
4th order linear	$6.0*10^{-6}$	$5.6*10^{-5}$
2nd order diagonal bilinear	$3.5*10^{-6}$	$2.9*10^{-5}$
3rd order diagonal bilinear	$3.0*10^{-6}$	$2.3*10^{-5}$
4th order diagonal bilinear	$2.7*10^{-6}$	$2.0*10^{-5}$
2nd order bilinear	$3.6*10^{-6}$	$2.9*10^{-5}$
3rd order bilinear	$3.0*10^{-6}$	$2.6*10^{-5}$
4th order bilinear	$2.6*10^{-6}$	$2.6*10^{-5}$

Table 8.1

The ability of the identified models to predict future column behaviour was tested by applying the input functions shown in Graph 8.5 to the experimental column and comparing the outputs with those predicted by the identified models. The results of this test for several different types of fourth order model are shown in Graphs 8.6 to 8.8.

Both the identification run and the prediction run were then repeated in order to find the source of the errors in the model predictions. Graphs 8.9 to 8.11 are the replications corresponding to Graphs 8.6 to 8.8, from which they show considerable deviation. This implies that the column parameters changed in between the two sets of runs.

8.4 DISCUSSION

The U-D algorithm again proved to be both reliable and robust, converging in all the identification runs attempted. Its resistance to process and measurement noise was demonstrated by its successful application to experimental data.

In general, the experimental results showed the same trends as the simulation in the previous chapter. This validated the approach of using a simulation to break down the task of distillation column identification into smaller steps. It also showed that the dynamic simulation itself was suitable for studies of this type.

Bilinear models were superior to linear models in their ability to fit experimental data, as shown in Graphs 8.2 to 8.4. The quality of fit of the bilinear models improved with increasing order, as shown by Table 8.1. Once again, the use of full bilinear models over diagonal bilinear models was not justified by their performance.

The variances shown in Table 8.1 are larger than those for the corresponding simulated results in Table 7.3, and do not show the same sized improvements for more complex models. The reason for this is that, with experimental data, the variances tabulated measure not only the quality of fit but also the noise present in the column measurements. Hence, a direct comparison between the variances generated with simulated and experimental data is misleading.

For the simulated study, the bilinear models were better at describing bottoms rather than distillate response. Graphs 8.3 and 8.4 show that a similar conclusion is not possible for the experimental study because of the obscuring effect of noise. By normal experimental standards, the quantity of noise is not great

and this highlights the quality of the fits obtained in this work.

The attempts to predict future column behaviour in Graphs 8.6 to 8.11 were considerably worse than for the simulated study. The improvement of using bilinear rather than linear models to predict future behaviour was insignificant in comparison with the remaining deviations. The differences between the two replications implies that this may have been due to the column parameters changing between the identification and prediction runs. There are a number of factors which had been assumed constant, but which may have changed at some stage. Table 8.2 shows the effect of a number of factors on the column outputs at an operating point in the centre of the region shown in Graph 8.1.

Factor Description	Change in Factor	Change in Column Outputs	
		Bottoms	Tops
Steam pressure	50kPa	0.0029 m.f.	0.0006 m.f.
Ambient temperature	10°C	-0.0189 m.f.	-0.0039 m.f.
Feed temperature	5°C	-0.0156 m.f.	-0.0029 m.f.

Table 8.2

The effects of quite feasible fluctuations in the feed and ambient temperatures were clearly significant in comparison with the normal operating region of the column. The effect due to steam pressure was calculated assuming the steam was dry and saturated, but if the steam condition varied then this effect too, would have been greater. The effect of these environmental factors demonstrates the importance of identification techniques, such as the U-D algorithm, which can be used in real-time to update the model parameters and controller tunings.

During the course of these experiments, it became clear that the experimental column still needed improvement in a number of areas. This was partially because this work looked much more closely at the column performance than previous works.

The experimental replication shown in Graph 8.9 deviated considerably from the original experiment shown in Graph 8.6. The deviation was particularly bad in the time interval from 360 to 480 minutes, when the reflux set point was increased to 0.831/min. This implied that either the controller or the reflux flow transducer were at fault. Experience suggests that the most likely cause was the unreliabilty of the flow sensors, which are operating well past their design capacity.

The feed composition was a major source of process noise and, while there was some degree of control of composition, it was insufficient to be able to set the composition to any predetermined value. The control of feed composition could be improved by the addition of a second feed pump and mixing two feed streams of different composition. The feed composition would be controlled by adjusting the ratio of the two flowrates.

The dominance of the column dynamics by the large reboiler, as mentioned in the previous chapter, could be decreased by changing the shape of the reboiler heads and the diameter of the associated pipework. It would be feasible to decrease the reboiler volume to 50% of its present volume.

The capacitance cells could be improved by increasing their resistance to impurities and ions in solution. This would require new cells with larger plate spacing and higher frequency oscillators, in the 50 to 100 MHz range. It would also be advantageous to change to a concentric cylinder cell layout because the parallel plate layout used in this work was reliant on the dimensional stability of the PVC block in which it was mounted.

CHAPTER 9

CONCLUSION

The aim of this work was to identify a non-linear model of a binary distillation column. The model had to be capable of describing the dynamics of a distillation column well enough for control applications over a wide operating region, but was to be sufficiently simple to allow for straightforward calculation of control action. The identification technique had to be robust and computationally inexpensive.

An experimental distillation column was upgraded and interfaced to the departmental VAX minicomputer, providing a facility for real-time data analysis and control. The modifications improved the measurements of column outputs, and the control of inputs.

The dynamics of the distillation column were simulated on the VAX minicomputer. This enabled the development of identification algorithms using simulated data, before applying them to experimental data and its associated difficulties. The quality of the fit, between the experimental column and its simulation, compared favourably with those of other researchers.

Distillation columns fall naturally into the class of non-linear systems termed bilinear, and the main effort of this work was directed at applying this class of model to distillation columns.

A case study of a simple bilinear system consisting of a simulated heated tank was undertaken because of the lack of previous work in this field and the complexity of distillation

columns. Bilinear models were more accurate than linear models in fitting the simulated system response. In addition, full bilinear models had no advantage over diagonal bilinear models, but required the identification of significantly more parameters. This study showed that there was considerable potential for the application of bilinear models to chemical processes, in particular, those in which the mass or energy balance equations have a purely bilinear structure.

Multi-input multi-output linear and bilinear models were fitted to both simulated and experimental column responses. The U-D algorithm used in these identification runs was both reliable and robust, converging for all runs attempted.

In the simulated study, bilinear models were shown to be superior to linear models in fitting and predicting column behaviour. Full bilinear models were shown to give little improvement over diagonal bilinear models. This was significant because diagonal bilinear models have much fewer parameters, especially for complex systems such as distillation columns, and they avoid the difficulties associated with over-parameterisation.

Two factors affected the trend in the column parameters; the turn-down of the column and the shifting of the composition profile along the column. The bilinear models were good at following changes due to the turn-down, but not the large fluctuations due to changes in the composition profile. This was consistent with the validity region of the model being a truncated cone in the state space, where the limitations were due to the physical constraints of the column, and the simplified liquid/vapour equilibrium relation inherent in the bilinear model.

The results for the experimental study were less conclusive. The bilinear models were better than linear models at fitting

experimental data, but there was no significant improvement in predicting future column behaviour. This was possibly due to the effect of environmental changes not included in the bilinear model. In previous works, these effects were not significant, but this work looked much more closely at the column performance than in the past.

REFERENCES

Akerlof, G., "Dielectric Constants of Some Organic Solvent-Water Mixtures". J.Am.Chem.Soc, v54, n11, p4125, 1932.

d'Alessandro, P., Isidori, A. and Ruberti, A., "Realisation and Structure Theory of Bilinear Dynamical Systems". SIAM Jour. Cont., v12, n3, Aug. 1974.

Astrom, K.J. and Wittenmark, B., "On Self-tuning Regulators". Automatica, v9, pp185-199, 1973.

Beghelli, S. and Guidorzi, R., "Bilinear Systems Identification from Input-Output Sequences". IVth IFAC Symposium on Identification and System Parameter Estimation, Tbilisi, Sept. 1976.

Benallou, A., Mellichamp, D.A. and Seborg, D.E., "Characterisation of Equilibrium Sets for Bilinear Systems with Feedback Control". Automatica, v19, n2, 1983.

Bierman, G.J., "Factorisation Methods for Discrete Sequential Estimation". Academic Press, 1977.

Bristol, E.H., "On a New Measure of Interaction for Multivariable Process Control". IEEE Trans. Auto. Cont, v11, n1, 1966.

Bruni, C., DiPillo, G. and Koch, G., "Bilinear Systems: An Appealing Class of Nearly Linear Systems in Theory and Application". IEEE Trans. Auto Cont, v19, n4, Aug. 1974.

Clarke, D.W. and Gawthrop, P.J., "Self-tuning Controller". Proc.IEE, v122, n9, 1975.

Dahlqvist, "Control of a Distillation Column Using Self-Tuning Regulators". Can.J.Chem.E., v59, Feb. 1981.

Defaye, G., Caralp, L., Jouve, P. and Baucou, R., "Modelisation d'un Depropaniseur". Automatisme, v22, n6, 1977.

Derese, I. and Noldus, E., "Design ofLinear Feedback Laws for Bilinear Systems". Int. Jour. Cont., v31, n2, pp219-237, 1980.

Dochain, D. and Bastin, G., "Adaptive Identification and Control Algorithms for Non-linear Bacterial Growth Systems". Automatica, v20, n5, pp621-634, 1984.

Elhennawey, M.S., Aidarous, S.E. and Shahein, H.I.H, "Identification of Bilinear Systems". Mathematics and Computers in Simulation XXIV, North-Holland, 1982.

Espana, M.D., "Modelisation Bilineaire de Colonnes a Distiller". Docteur Ingenieur Thesis, Institut National Polytechnique de Grenoble, Grenoble, 1977.

Espana, M.D. and Landau, I.D., "Bilinear Approximation of the Distillation Processes". Ricerche Di Automatica, v6, n1, July 1975.

Fortescue, T.R., Kershenbaum, L.S. and Ydstie, B.E., "Implementation of Self Tuning Regulators with Variable Forgetting Factors". Automatica, v17, pp831-835, 1981.

Funahashi, Y., "An Observable Canonical Form of Discrete-Time Bilinear Systems". IEEE Trans. Auto. Cont., v24, n5, Oct. 1979.

Gallun, S.E. and Holland, C.D., "Gear's Procedure for the Simultaneous Solution of Differential and Algebraic Equations with Application to Unsteady State Distillation Problems". Comp. and Chem. Eng., v6, n3, 1982.

Gear, C.W., "The Automatic Integration of Ordinary Differential Equations". Commun. ACM, v14, n3, 1971a.

Gear, C.W., "Simultaneous Solution of Differential-algebraic Equations". IEEE Trans. Circuit Theory, v18, n1, 1971b.

Goodwin, G.C., McInnis, B. and Long, R.S., "Adaptive Control Algorithms for Waste Water Treatment and pH Neutralisation" Technical Report EE8112, University of Newcastle, Australia 1981.

Goodwin, G.C. and Sin, K.S., "Adaptive Filtering Prediction and Control". Prentice-Hall, 1984.

Graupe, D., "Identification of Systems". Van Nostrand Reinhold, 1972.

Gustavsson, I., "Survey of Applications of Identification in Chemical and Physical Processes". Automatica, v11, pp.3-24 1975.

Hindmarsh, A.C., "LSODE and LSODI, Two New Initial Value Ordinary Differential Equation Solvers". ACM Signum Newsletter, v15 n4, pp.10-11, 1980.

Holland, C.D. and Liapis, Y., "Computer Methods for Solving Dynamic Separation Problems". McGraw Hill, 1983.

Huckaba, C.E., Franke, F.R., May, F.P., Fairchild, B.T. and Distefano, G.P., "Experimental Confirmation of a Predictive Model for Dynamic Distillation". Chem. Eng. Progr. Symp. Ser. v61, n55, 1963.

Hsia, T.C., "System Identification". Lexington Books, 1977.

Inagaki, M., "On-line Parameter Identification of Bilinear Systems". IEEE Trans. Auto. Cont., v27, n4, Aug. 1982.

Ionescu, T. and Monopoli, R.V., "On Stabilisation of Bilinear Systems via Hyperstability". IEEE Trans. Auto. Cont., v20, n2, 1975.

Isermann, R., "Practical Aspects of Process Identification". Automatica, v16, pp.577-579, 1980.

Isermann, R., "Digital Control Systems". Springer-Verlag, 1981.

Isermann, R., Baur, U., Bamberger, W., Kneppo, P. and Siebert, H., "Comparison of Six On-Line Identification and Parameter Estimation Methods". Automatica, v10, pp.81-103, 1974.

Jan, Y.G. and Wong, K.M., "Bilinear System Identification by Block Pulse Functions". Journal of Franklin Institute, v312 n5, pp349-359, Nov. 1981.

Kailath, T., "Linear Systems". Prentice-Hall, Englewood Cliffs, 1980.

Kubrusly, C.S., "Identification of Discrete-time Stochastic Bilinear Systems". Int. Jour. Cont., v33, n2, pp291-309, 1981.

Kumar, S., Wright, J.D. and Taylor, P.A., "Modelling and Dynamics of an Extractive Distillation Column". Can.J.Chem.Eng., v62, Dec, 1984.

Ljung, L. and Soderstrom, T., "Theory and Practice of Recursive Identification". MIT Press, 1983.

Mehra, R.K., "Choice of Input Signals". Trends and Progress in System Identification, Ed. Eykhoff, Pergamon Press, 1981.

Michelsen, M.L., "An Efficient General Purpose Method for the Integration of Stiff Ordinary Differential Equations". A.I.Chem.E. Jour., v22, p594, 1976.

Mohler, R.R., "Bilinear Control Processes". Academic Press, 1973

Morris, A.J., Nazer, Y. and Wood, R.K., "Single and Multivariable Application of Self-Tuning Controllers". IEE Control Engineering Series 15, 1981.

Morris, C.G. and Svrcek, W.Y., "Dynamic Simulation of Multicomponent Distillation". Can.J.Chem.Eng., v59, June, 1981.

Ohkawa, F. and Yonezawa, Y., "A Model Reference Adaptive Control System for Discrete Bilinear Systems". Int.J.Cont., v37, n5, 1983.

Rao, K.V., Frick, P.A. and Mohler, R.R., "On Bilinear System Identification by Walsh Functions". IVth IFAC Symposium on Identification and System Parameter Estimation, Tbilisi, Sept. 1976.

Saridis, G.N., "Comparison of Six On-Line Identification Algorithms". Automatica, v10, pp.69-79, 1974.

Saridis, G.N. and Stein G., "A New Algorithm for Linear System Identification". IEEE Trans. Auto. Cont., v13, n5, 1968.

Sastry, V.A., Seborg, D.E. and Wood, R.K., "Self-tuning Regulator Applied to a Binary Distillation Column". Automatica, v13, pp.417-424, 1977.

Shinsky, F.G., "Process Control Systems". McGraw-Hill, 1967.

Smith, B.D., "Design of Equilibrium Stage Processes". McGraw-Hill, 1963.

Strejc, V., "Least Squares Parameter Estimation". Automatica, v11, pp.535-550, 1980.

Subba Rao, T. and Gabr, M.M., "An Introduction to Bispectral Analysis and Bilinear Time Series Models". Lecture Notes in Statistics Series, v24, Springer-Verlag, 1984.

Svoronos, S., Stephanopoulos, G. and Aris, R., "Bilinear Approximation of General Non-linear Dynamic Systems with Linear Inputs". Int.J.Cont., v31, n1, 1980.

Svoronos, S., Stephanopoulos, G. and Aris, R., "On Bilinear Estimation and Control". Int.J.Cont., v34, n4, 1981.

Svrcek, W.Y., PhD Thesis, University of Alberta, Edmonton, 1967.

Tyreus, B.D., Luyben, W.L. and Schlesser, W.E., "Stiffness in Distillation Models and the Use of an Implicit Integration Method to Reduce Computation Times". Ind. Eng. Chem., Proc. Des. Dev., v14, n4, 1975.

Webb, D. and Edgar, T.F., "Decoupling Controller Simplification via Model Reduction". ISA Trans., v18, n4, 1979.

Williamson, D., "Observation of Bilinear Systems with Application to Biological Control". Automatica, v13, pp.243-254, 1977.

Wilson, G., "Distillation Column Dynamics and Control". PhD Thesis, University of Canterbury, 1979.

Zarrop, M.B., "Optimal Experiment Design for Dynamic System Identification". Lecture Notes in Control and Information Sciences, v21, Springer-Verlag, 1979.

APPENDIX A

MICROPROCESSOR SOFTWARE

A.1 PR9MON COMMANDS

PR9MON is a monitor for an AMI prototyping board upgraded to use a Motorola 6809 microprocessor. It is based on Percom's PSYMON but has been modified to support local hardware and to simplify the downloading of Motorola hex tape format files from the VAX.

The following is a summary of the PR9MON commands that can be typed from the terminal when prompted by "CMD?". A detailed description of all commands, except load and transparent, can be found in the PSYMON user's guide.

M [address] - memory examine/change

G [address] - go to address

R [register] - register examine/change

L - load program from VAX

S [start] [end] - save program to VAX

B [address] - set/list breakpoints

U [address] - unset breakpoints

T - enter transparent mode, to exit type ESC

Z - jump to eprom at address $3000

The following procedure will download the program COLSYS.LDA from the VAX :

1. Enter transparent mode by typing "T".
2. Log onto VAX and type "TYPE COLSYS.LDA", but not CR.
3. Exit transparent mode by hitting ESC.
4. Commence downloading by typing "L".

A.2 COLSYS INSTRUCTION SET

The microprocessor is usually transparent to characters passing between the VAX and the operator, except for the CONTROL/P character, which is always intercepted, and instructions to COLSYS, which are intercepted when the microprocessor is not in PASS ALL mode. Instructions to COLSYS are strings beginning with the hash character "#", and finishing with a carriage return, shown <CR>. If an instruction results in the microprocessor giving a response, this is directed either to the operator or to the VAX, depending on where the instruction originated.

There are times when real-time programs are not running and the hash should be ignored by the microprocessor; for example, if a source file is to be listed on the terminal, and it contains a hash character then the COLSYS will interpret the hash as it is being listed, and try to perform some instruction. To prevent this, hitting CONTROL/P at the terminal puts the microprocessor into PASS ALL mode, in which the special significance of the hash character is ignored. The mode is indicated by the front panel PASS ALL light. For real-time programs, this mode can be turned of with a second CONTROL/P.

The following is a list of all the possible instructions to COLSYS :

#R<CR> - Report input channels

This command string instructs the micropocessor to report the values of the input channels. The report format is:

#I,[ch 1],[ch 2],...,[ch 10]<CR>

#I are two header characters, which may be ignored by the user, but are used by COLSYS to eliminate echoing of reports to the VAX on the operators terminal. The data items are separated by commas as this allows for unformatted high level

language reads on the VAX. Each number will be a decimal integer unless a #H instruction has been issued.

The ten input channels presently selected are :

No		Description
1	-	Tray 1 temperature, range 0 to 4095 => 60 to 100°C
2	-	Tray 8 temperature, range 0 to 4095 => 60 to 100°C
3	-	Distillate flow sensor frequency
4	-	Bottoms flow sensor frequency
5	-	Cell 1 temperature, range 0 to 4095 => 0 to 100°C
6	-	Cell 2 temperature, range 0 to 4095 => 0 to 100°C
7	-	Cell 3 temperature, range 0 to 4095 => 0 to 100°C
8	-	Capacitance cell 1 oscillator frequency
9	-	Capacitance cell 2 oscillator frequency
10	-	Capacitance cell 3 oscillator frequency

#A [FREQ]<CR> - Automatically report input channels

where [FREQ] is in the range 1 to 255 seconds

This command string instructs COLSYS to repeatedly sample and report the values of the input channels at the specified frequency. The report format is the same as that for the #R instruction. If the command string is simply #A<CR> then automatic reporting is turned off, as it will be if CONTROL/C or CONTROL/Y is hit at the terminal.

#C [LOOP] [SP]<CR> - Change controller set point

where [LOOP] options are F,R or S

and [SP] is in the range 0 to 65535 for [LOOP] = F or R

and 0 to 255 for [LOOP] = S

This instruction changes the set point for either the feed, reflux or steam flowrate control loop. The set points are expected to be decimal integers unless special steps are taken. Data items can be separated by commas or blanks.

#E [Y/N]<CR> - Echo enable/disable

This command switches on or off the echoing of reports to the VAX on the user terminal. On reset, the default is echo on.

#H [Y/N]<CR> - Hexadecimal enable/disable

This command string sets the integer format used by COLSYS to either hexadecimal or decimal number base. If the high level language program running on the VAX is in FORTRAN you have the option of using hexadecimal numbers directly, with the Z descriptor in format statements. The hexadecimal format is slightly more compact and hence faster. On reset and power on, the default is decimal.

#W [Y/N]<CR> - Warning enable/disable

This command sets whether the microprocessor checks liquid levels and product temperatures to see if they are within prespecified limits. If an error condition exists a warning message is sent to the operators terminal every 10 seconds.

#F [mode]<CR> - Feed mixing

where [mode] is in the range 0 to 3

In mode 0 no feed mixing is performed and all other outstanding feed mixing requests are cancelled upon receipt of this instruction.

In mode 1 the microprocessor attempts to control the composition in the feed tanks at 50 mole % of methanol. This is achieved by an on/off type controller which switches the solenoids from the products tanks to the feed tanks.

Mode 2 is used if a feed of constant composition, but not necessarily 50 %, is required. It opens all the valves from the product tanks to the feed tanks at regular

intervals. It is necessary because if the valves are left permanently open the two feed tanks tend to have different compositions.

Mode 3 initiates a mixing sequence in order to obtain different compositions in the two feed tanks. The valve from the distillate tank to feed tank 0 and the valve from the bottoms tank to feed tank 1 are left permanently open. The two other valves at the outlets of the products tanks are only open for [ratio]/([ratio]+1) fraction of the time. This results in feed tank 0 always having a higher concentration of the more volatile component, the difference being greater for small [ratio]. The sequence is terminated when both products tanks are empty and the feed mixing then reverts to mode 0.

#V<CR> - To display the setting of the tank solenoids

#V [byte]<CR> - To change the settng of the tank solenoids

where [byte] is in the range 0 to 3F in hexadecimal and each of the six bits in the number switches one of the solenoid valves.

#P<CR> - To display all pump settings

#P [number]<CR> - To display one pump setting

#P [number] [out]<CR> - To change one pump setting

where [NUMBER] = 1 - feed
2 - reflux
3 - distillate
4 - bottoms
5 - steam

and [out] is a number in the range 0 to 255

#X [ch no]<CR> - Select sample channel at solenoid manifold

where [ch no] = 0 - distillate
1 - bottoms
2 - feed
3 - reference

#T<CR> - Report storage tank volumes as a percentage of full.

#Z<CR> - Shut down column

A procedure to shut the column down in a controlled manner is initiated. The operator is given the steps to follow in the appropriate order.

#D [add] [bytes]<CR> - Display memory

where [add] is in the range 0 to FFFF in hexadecimal
and [bytes] is in the range 1 to FF in hexadecimal

This instruction is included for debugging purposes only. It displays the contents of the microprocessor memory starting at [add] and continuing for [bytes] number of bytes. If the instruction is not fully specified then the values from the previous #D instruction are used.

#S [add] [byte 1] [byte 2]...[byte n]<CR> - Set memory

where [add] is in the range 0 to FFFF in hexadecimal
and [byte ?] is in the range 1 to FF in hexadecimal

Again this instruction is included for debugging purposes only. It sets the contents of the microprocessor memory at [add] to [byte 1] and continues filling n successive memory locations with the data provided.

A.3 CONTROL LOOPS

The column operating program COLSYS implemented four low level control loops. These all used the digital equivalent of a conventional PI controller, with a sampling time of 1 second. The following is the algorithm used:

1. $E_{new} = SP - PV$
2. $\Delta V = K_1 * E_{new} + K_2 * E_{old}$
3. Limit ΔV such that: $\Delta V \leq S_{up}$ if $\Delta V > 0$
   $\Delta V \geq S_{down}$ if $\Delta V < 0$
4. $V_{new} = V_{old} + \Delta V$
5. Limit V_{new} to the range: $0 \leq V_{new} \leq 4095$
6. $V_{out} = V_{new} / 16$

where SP is the controller set point

PV is the process variable

ΔV is the change in valve position

K_1 and K_2 are the controller parameters

S_{up} and S_{down} are the slew rate limits (all set to 160)

E_{new} is the new error in the process variable

E_{old} is the previous error in the process variable

V_{new} is the new valve position

V_{out} is the controller output in D/A units

Measured Variable	Manipulated Variable	K_1	K_2
Feed flowrate	Feed pump	3.6	-3.0
Reflux flowrate	Reflux pump	3.6	-3.0
Reflux level	Tops pump	-10.06	10.0
Reboiler level	Bottoms pump	-10.03	10.0

CONTROLLER PARAMETERS

A.4 FILTERS

The column operating program COLSYS implemented four first order filters. The following digital algorithm was used with a sampling time of 1 second :

$$Y_{new} = Y_{old} + (U_{new} - Y_{new}) * \Delta T / \tau$$

where U_{new} is the new filter input

Y_{old} is the old filter output

Y_{new} is the new filter output

ΔT is the sampling time in seconds

τ is the filter time constant in seconds

The following measurements where filtered, all with a time constant of 10 seconds :

1. Temperature on tray 1
2. Temperature on tray 8
3. Tops flowrate
4. Bottoms flowrate

A.5 ASSEMBLER LANGUAGE PROGRAM LISTINGS

A.5.1 PR9MON MONITOR

```
         NAM      PR9MON
         OPT      NOG,LLE=120,P=62
*
* A MONITOR FOR AN AMI PROTO BOARD UPGRADED TO
* USE A MOTOROLA 6809 MICROPROCESSOR. IT IS BASED
* ON PERCOM'S PYSMON AS MODIFIED BY P.W.M.JANSSEN.
*
* COMMANDS:
*   M <ADDRESS> - MEMORY EXAMINE/CHANGE
*   G <ADDRESS> - GO TO ADDRESS
*   R <REGISTER> - REGISTER EXAMINE/CHANGE
*   L - LOAD PROGRAM FROM TAPE
*   S <START> <END> - SAVE PROGRAM TO TAPE
*   B <ADDRESS> - SET/LIST BREAKPOINTS
*   U <ADDRESS> - UNSET BREAKPOINTS
*   T - ENTER TRANSPARENT MODE
*   Z - JUMP TO EPROM AT ADDRESS $3000
*
* CALLABLE SUBROUTINES:
*   INCHR - INPUT CHARACTER FROM CONSOLE
*   OUTCHR - OUTPUT CHARACTER TO CONSOLE
*   REQIO - PERFORM I/O TO PERIPHERAL
*   GETHEX - INPUT HEX NUMBER FROM CONSOLE
*   INHEX - INPUT HEX DIGIT FROM CONSOLE
*   DSPSBY - DISPLAY SINGLE BYTE & SPACE
*   DSPDBY - DISPLAY DOUBLE BYTE & SPACE
*   OUTHEX - DISPLAY 2 HEX DIGITS
*   PSTRNG - DISPLAY STRING ON CONSOLE
*   LOAD - LOAD HEX PROGRAM FROM CONSOLE
*   SAVE - SAVE HEX PROGRAM TO CONSOLE
*   CRLF - BEGIN NEW LINE ON CONSOLE
*   OUTS - OUTPUT SPACE TO CONSOLE
*   NULL - OUTPUT 4 NULLS TO CONSOLE
*
* ALL I/O WITHIN PR9MON IS DONE THROUGH THE
* USE OF DEVICE CONTROL BLOCKS. THIS ALLOWS
* EASY MODIFICATION BY THE USER. PR9MON HAS
* FOUR DCB POINTERS INITIALISED TO POINT TO
* THE CONSOLE (ACIA) DCB. THEY ARE USED AS
* FOLLOWS:
*   CIDCB - POINTS TO DCB USED FOR COSOLE
*           INPUT (CHARACTER I/O).
*   CEDCB - POINTS TO DCB USED FOR ECHO OF
*           CHARACTERS RECEIVED USING CIDCB.
*           ECHO MAY BE SUPPRESSED BY SETTING
*           THIS POINTER TO ZERO.
*   CODCB - POINTS TO DCB USED FOR CONSOLE
*           OUTPUT (CHARACTER I/O).
*   TPDCB - POINTS TO DCB USED FOR PR9MON
*           TAPE LOAD & SAVE COMMANDS.
*
*
* SYSTEM ADDRESS CONSTANTS
TERMNL   EQU      $FBCE    TERMINAL ACIA
REMOTE   EQU      $DFB8    REMOTE ACIA
```

```
* D.A.M MODULE
DMSB    EQU     $FBC8
DCSR1   EQU     DMSB+1
DLSB    EQU     DCSR1+1
DCSR2   EQU     DLSB+1
* D/A PIAS
V5      EQU     $DF90
V4      EQU     $FBC0
V3      EQU     V4+2
V2      EQU     V3+2
V1      EQU     V2+2
V5CSR   EQU     V5+1
V4CSR   EQU     V4+1
V3CSR   EQU     V3+1
V2CSR   EQU     V2+1
V1CSR   EQU     V1+1
* SOLENOID VALVES PIA
SOLV    EQU     $DF92
SOLVCS  EQU     SOLV+1
* ASCII CHARACTER CONSTANTS
CR      EQU     $0D     CARRIAGE RETURN
LF      EQU     $0A     LINE FEED
SP      EQU     $20     SPACE
* ACIA CONTROL CONFIGURATIONS
RESET   EQU     $03     RESET ACIA
CONFIG  EQU     $01     7 DATA, 2 STOP, EVEN PARITY
RDRON   EQU     CONFIG+$40      READER ON (RTS HIGH)
RDROFF  EQU     CONFIG  READER OFF (RTS LOW)
* DCB OFFSETS
DCBLNK  EQU     0       PTR TO NEXT DCB IN CHAIN
DCBDID  EQU     2       ASCII 2 CHAR DEVICE ID
DCBDVR  EQU     4       DEVICE DRIVER ADDRESS
DCBIOA  EQU     6       DEVICE I/O ADDRESS
DCBERR  EQU     8       ERROR STATUS CODE
DCBEXT  EQU     9       DCB EXTENSION BYTE COUNT
DCBAPP  EQU     10      DRIVER DCB APPENDAGE
* DCB FUNCTION CODES
READFN  EQU     $01     READ FUNCTION CODE
WRITFN  EQU     $02     WRITE FUNCTION CODE
STATFN  EQU     $04     STATUS FUNCTION CODE
CNTRFN  EQU     $08     DEVICE CONTROL FN CODE
* DATA BASE DEFINITIONS
        ORG     $FC00   START OF UPPER RAM
        RMB     $390    USABLE STACK SPACE
* INTERNAL STACK (& REGISTER) SPACE
STACK   EQU     *
REGC    RMB     1       CONDITION CODE REGISTER
REGA    RMB     1       A REGISTER
REGB    RMB     1       B REGISTER
REGD    RMB     1       DIRECT PAGE REGISTER
REGX    RMB     2       X REGISTER
REGY    RMB     2       Y REGISTER
REGU    RMB     2       U STACK POINTER
REGP    RMB     2       PROGRAM COUNTER
* BREAKPOINT TABLE
BPTABL  RMB     30      SPACE FOR 10 BREAKPOINTS
BPTEND  EQU     *
* WORK AREAS
MEMPTR  RMB     2       MEMORY POINTER FOR 'M' CMD
USRTBL  RMB     2       ADDRESS OF USER COMMAND TBL
```

```
COMAND  RMB     1       COMMAND CHAR STORAGE
CKSUM   RMB     1       LOAD, SAVE CHECKSUM
BEGADD  RMB     2       BEGIN ADDRESS FOR SAVE
ENDADD  RMB     2       END ADDRESS FOR SAVE
STKPTR  RMB     2       CONTENTS OF STACK PTR
* CONSOLE DCB
CONDCB  RMB     10      NO EXTENSIONS
* REMOTE DCB
REMDCB  RMB     10
* DCB POINTERS
DCBCHN  RMB     2       BASE OF DCB CHAIN
CIDCB   RMB     2       CONSOLE INPUT DCB
CEDCB   RMB     2       CONSOLE ECHO DCB
CODCB   RMB     2       CONSOLE OUTPUT DCB
TPDCB   RMB     2       TAPE DCB
* INTERUPT & RESET VECTORS
        ORG     $FFF2
SWI3V   RMB     2       SOFTWARE INTERUPT 3
SWI2V   RMB     2       SOFTWARE INTERUPT 2
FIRQV   RMB     2       FAST INTERUPT REQUEST
IRQV    RMB     2       INTERUPT REQUEST
SWIV    RMB     2       SOFTWARE INTERUPT
NMIV    RMB     2       NON-MASKABLE INTERUPT
RESTRT  RMB     2       RE-ENTRY INTO PR9MON
*********************************************************
* START OF 2532 EPROM ON RAM/EPROM BOARD
*********************************************************
        ORG     $2000
* INITIALISATION UPON RESET
INIT    LDS     #STACK          SET UP STACK POINTER
        TFR     S,X             POINT X AT STACK
INIT1   CLR     ,X+             CLEAR A BYTE
        CMPX    #CONDCB+2       ALL FIELDS CLEAR?
        BNE     INIT1           LOOP IF NOT
        LDY     #RAMINT         SET UP DCB RAM
INIT2   LDD     ,Y++            MOVE 2 BYTES
        STD     ,X++
        CMPX    #TPDCB+2        END OF DCB RAM?
        BNE     INIT2           LOOP IF NOT
        LDX     #SWI3V          SET UP INTERUPT VECTORS
INIT3   LDD     ,Y++
        STD     ,X++
        CMPX    #RESTRT
        BNE     INIT3
* ACIA INITIALISATION
        LDX     #CONDCB         POINT TO DCB
        LDD     #$0308          A=RESET,B=CNTL
        JSR     REQIO           MASTER RESET
        LDA     #CONFIG         CONFIGURE ACIA
        JSR     REQIO
        LDX     #REMDCB         REPEAT FOR REMOTE
        LDD     #$0308
        JSR     REQIO
        LDA     #CONFIG
        JSR     REQIO
* PIA INITIALISATION, ASSUMES RESET HAS CLEARED REGS
CONPIA  CLRA
        STA     DMSB    SET PIAS FOR INPUT
        STA     DLSB
        COMA
```

```
         STA     V1      SET PIAS FOR OUTPUT
         STA     V2
         STA     V3
         STA     V4
         STA     V5
         STA     SOLV
*
         LDA     #$36
         LDB     #6
         LDX     #V4CSR
CON1     STA     ,X++    SET CONTROL REGISTERS
         DECB
         BNE     CON1
         STA     V5CSR
         STA     SOLVCS
*********************************************************
* USER ENTRY POINT
*********************************************************
MONENT   LDS     #STACK  SET STACK POINTER
         STS     STKPTR
*********************************************************
* GET COMMAND
*********************************************************
GETCMD   LDX     #PROMPT DISPLAY PROMPT
         JSR     PSTRNG
         JSR     INCHR   INPUT COMMAND CHARACTER
         BSR     LOOKUP  LOOK IT UP
         BNE     GETCMD  LOOP IF NOT FOUND
         JSR     OUTSP   OUTPUT A SPACE
         JSR     [,X]    CALL COMMAND ROUTINE
         BRA     GETCMD  GO BACK FOR MORE
PROMPT   FCB     CR,LF,0,0,0,0
         FCC     'CMD'
         FCB     '?+$80  END OF STRING
*********************************************************
* LOOK UP COMMAND IN TABLE
*********************************************************
LOOKUP   LDY     #COMAND         POINT Y TO COMMAND
         STA     ,Y              SAVE COMMAND CHARACTER
         LDX     USRTBL          GET USER TABLE ADDRESS
         BEQ     LOOK1           GO IF NONE
         BSR     SEARCH          SEARCH USER TABLE
         BEQ     SERCHX          GO IF FOUND
LOOK1    LDX     #CMDTBL         SEARCH INTERNAL TABLE
*********************************************************
* GENERAL TABLE SEARCH
*
* ENTRY REQUIREMENTS:   X - POINTS TO TABLE
*                       Y - POINTS TO ITEM
*                       FIRST BYTE OF TABLE MUST
*                       CONTAIN ITEM LENGTH
*                       LAST BYTE MUST BE FF
*
* EXIT CONDITIONS:      C - Z SET IF FOUND, CLEAR
*                           IF NOT FOUND
*                       X - POINTS TO ADDRESS OF
*                           ROUTINE FOR MATCH
*                       A,B - CHANGED
*
*********************************************************
```

```
SEARCH  LDB     ,X+     GET ITEM LENGTH
SERCH1  BSR     COMPAR  COMPARE CURRENT ITEM
        ABX             ADVANCE TO NEXT ITEM
        BEQ     SERCHX  EXIT IF MATCH
        LEAX    2,X     STEP OVER ADDRESS
        TST     ,X      END OF TABLE?
        BPL     SERCH1  LOOP IF NOT
SERCHX  RTS
*********************************************************
* GENERAL STRING COMPARE
*
* ENTRY REQUIREMENTS:   X - ADDRESS OF STRING 1
*                       Y - ADDRESS OF STRING 2
*                       B - LENGTH OF STRINGS
*
* EXIT CONDITIONS:      C - SET PER COMPARE 1:2
*                       B,X,Y - UNCHANGED
*                       A - CHANGED
*
*********************************************************
COMPAR  PSHS    B,X,Y   SAVE REGISTERS
COMP1   LDA     ,X+     GET NEXT CHARACTER
        CMPA    ,Y+     COMPARE IT
        BNE     COMP2   EXIT IF UNMATCHED
        DECB            DECREMENT LOOP COUNT
        BNE     COMP1
COMP2   PULS    B,X,Y,PC RESTORE REGISTERS & EXIT
*********************************************************
* LOAD PROGRAM FROM TAPE
*********************************************************
TLOAD   LDD     CIDCB   SAVE CONSOLE DCBS
        LDX     CEDCB
        PSHS    A,B,X
        LDX     TPDCB   POINT TO TAPE DCB
        CLRA            SET D TO 0
        CLRB
        STX     CIDCB   SET TAPE IN, NO ECHO
        STD     CEDCB
        LDA     #CR     SEND <CR> TO START DOWNLOAD
        LDB     #WRITFN
        JSR     REQIO
        BSR     LOAD    LOAD THE TAPE
        PULS    A,B,X   RESTORE CONSOLE DCBS
        STD     CIDCB
        STX     CEDCB
        TST     CKSUM   ANY ERRORS?
        BEQ     LOADX   GO IF NOT
*********************************************************
* DISPLAY ERROR INDICATOR OF '?'
*********************************************************
ERROR   LDA     #'?     DISPLAY ERROR INDICATOR
        JMP     OUTCHR
*********************************************************
* LOAD PROGRAM IN HEX FORMAT
*
* ENTRY REQUIREMENTS:   NONE
*
* EXIT CONDITIONS:      ALL REGISTERS CHANGED
*                       CHKSUM NON-ZERO IF ERROR
*********************************************************
```

```
LOAD    TFR     S,Y     MARK STK FOR ERROR RCVRY
LOAD1   JSR     INCHR   GET A CHARACTER
LOAD2   CMPA    #'S     START OF RECORD?
        BNE     LOAD1   LOOP IF NOT
        JSR     INCHR   GET ANOTHER CHARACTER
        CMPA    #'9     END OF LOAD?
        BEQ     LOADX   GO IF YES
        CMPA    #'1     START OF RECORD?
        BNE     LOAD2   LOOP IF NOT
        CLR     CKSUM   INIT CHECKSUM
        BSR     INBYTE  READ LENGTH
        SUBA    #2      ADJUST IT
        TFR     A,B     SAVE IN B
        BSR     INBYTE  GET ADDRESS HI
        STA     ,--S    SAVE ON STACK
        BSR     INBYTE  GET ADDRESS LO
        STA     1,S     PUT ON STACK
        PULS    X       ADDRESS NOW IN X
LOAD3   BSR     INBYTE  READ A BYTE
        DECB            DCREMENT COUNT
        BEQ     LOAD4   GO IF DONE
        STA     ,X      STORE BYTE
        CMPA    ,X+     VERIFY GOOD STORE
        BNE     LOAD5   GO IF ERROR
        BRA     LOAD3
LOAD4   INC     CKSUM   CHECK CHECKSUM
        BEQ     LOAD1   LOOP IF GOOD
LOAD5   LDA     #$FF    SET ERROR FLAG
        STA     CKSUM
        TFR     Y,S     RESTORE STACK
LOADX   RTS
*********************************************************
* INPUT BYTE
*********************************************************
INBYTE  BSR     INHEX   GET HEX DIGIT
        BEQ     LOAD4   GO IF ERROR
        ASLA            SHIFT TO MS HALF
        ASLA
        ASLA
        ASLA
        PSHS    A       SAVE DIGIT
        BSR     INHEX   GET ANOTHER DIGIT
        BEQ     LOAD4   GO IF ERROR
        ADDA    ,S      COMBINE HALVES
        STA     ,S      SAVE ON STACK
        ADDA    CKSUM   ADD TO CHECKSUM
        STA     CKSUM
        PULS    A,PC    GET RESULT & RETURN
*********************************************************
* GET HEX NUMBER FROM CONSOLE
*
* ENTRY REQUIREMENTS:   NONE
*
* EXIT REQUIREMENTS:    A - LAST CHAR INPUT
*                       B - HEX DIGIT COUNT
*                       X - HEX NUMBER
*                       C - SET ACCORDING TO B
*
*********************************************************
```

```
GETHEX  CLRB            INIT DIGIT COUNT, RESULT
        LDX     #0
GETHX1  BSR     INHEX   GET A DIGIT
        BEQ     GETHX2  GO IF NOT HEX
        EXG     D,X     OLD RESULT TO A,B
        ASLB            SHIFT LEFT 1 DIGIT
        ROLA
        ASLB
        ROLA
        ASLB
        ROLA
        ASLB
        ROLA
        EXG     D,X     REPLACE RESULT
        LEAX    A,X     ADD IN NEW DIGIT
        INCB            ADD TO DIGIT COUNT
        BRA     GETHX1  LOOP FOR MORE
GETHX2  TSTB            SET/RESET Z FLAG
        RTS
*********************************************************
* GET HEX DIGIT FROM CONSOLE
*
* ENTRY REQUIREMENTS:   NONE
*
* EXIT CONDITIONS:      A - HEX DIGIT OR NON-HEX
*                       C - Z FLAG SET IF NOT HEX
*                       ALL OTHER REGISTERS PRESERVED
*
*********************************************************
INHEX   BSR     INCHR   GET A CHARACTER
        PSHS    A       SAVE IT
        SUBA    #$30    CONVERT TO BINARY
        BMI     INHEX2  GO IF NOT NUMERIC
        CMPA    #$09    GREATER THAN 9?
        BLS     INHEX1  GO IF NOT
        SUBA    #$07    CONVERT LETTER
INHEX1  CMPA    #$0F    GREATER THAN 15?
        BLS     INHEX3  GO IF NOT
INHEX2  LDA     ,S      GET ORIGINAL CHAR BACK
INHEX3  CMPA    ,S+     SET/RESET Z FLAG
        RTS
*********************************************************
* CONSOLE INPUT ROUTINE
*
* ENTRY REQUIREMENTS:   NONE
*
* EXIT CONDITIONS:      A - CHARACTER WITH PARITY
*                           REMOVED
*
*********************************************************
INCHR   PSHS    B,X     SAVE REGISTERS
        LDX     CIDCB   POINT TO INPUT DCB
        LDB     #READFN SET UP FOR READ
        BSR     REQIO   READ A CHARACTER
        ANDA    #$7F    REMOVE PARITY
        LDX     CEDCB   POINT TO ECHO DCB
        PSHS    A       SAVE CHARACTER
        BNE     OUTCH1  GO IF ECHO
        PULS    A,B,X,PC  RESTORE & RETURN
```

```
*************************************************************
* CONSOLE OUTPUT ROUTINE
*
* ENTRY REQUIREMENTS:   A - CHARACTER TO BE
*                           OUTPUT TO CONSOLE
*
* EXIT CONDITIONS:      ALL REGISTERS PRESERVED
*                       EXCEPT C
*
*************************************************************
OUTCHR  PSHS    A,B,X   SAVE REGISTERS
        LDX     CODCB   POINT TO OUTPUT DCB
OUTCH1  LDB     #WRITFN SET FUNCTION
        BSR     REQIO   OUTPUT THE CHARACTER
        PULS    A,B,X,PC  RESTORE & RETURN
*************************************************************
* PERFORM I/O REQUESTS
*
* ENTRY REQUIREMENTS:   A - DRIVER PARAMETER
*                       B - FUNCTION CODE
*                       X - DCB ADDRESS
*
* EXIT CONDITIONS:      A - DRIVER RESULT
*                       ALL OTHERS PRESERVED EXCEPT C
*
*************************************************************
REQIO   PSHS    B,X,Y       SAVE REGISTERS
        JSR     [DCBDVR,X]  CALL DRIVER
        PULS    B,X,Y,PC    RESTORE & RETURN
*************************************************************
* DISPLAY DOUBLE BYTE
*
* ENTRY REQUIREMENTS:   A,B - DOUBLE BYTE TO BE PRINTED
*
* EXIT CONDITIONS:      ALL REGISTERS PRESERVED
*                       EXCEPT C
*
*************************************************************
DSPDBY  BSR     OUTHEX  DISPLAY A AS 2 HEX DIGITS
        EXG     A,B     LS BYTE TO A
        BSR     DSPSBY  DISPLAY AS 2 DIGITS, SPACE
        EXG     A,B     RESTORE A & B
        RTS
*************************************************************
* DISPLAY A BYTE AND SPACE
*
* ENTRY REQUIREMENTS:   A - BYTE TO BE DISPLAYED
*
* EXIT CONDITIONS:      ALL REGISTERS PRESERVED
*                       EXCEPT C
*************************************************************
DSPSBY  BSR     OUTHEX  DISPLAY BYTE IN A
*************************************************************
* OUTPUT A SPACE TO THE CONSOLE
*
* ENTRY REQUIREMENTS:   NONE
*
* EXIT CONDITIONS:      ALL REGISTERS PRESERVED
*                       EXCEPT C
*************************************************************
```

```
OUTSP   PSHS    A       SAVE A REGISTER
        LDA     #SP     OUTPUT A SPACE
*********************************************************
* OUTPUT CHARACTER, RESTORE A, & RETURN
*********************************************************
OUTCHX  BSR     OUTCHR  DISPLAY CHARACTER
        PULS    A,PC    RESTORE & EXIT
*********************************************************
* DISPLAY A REGISTER AS 2 HEX DIGITS
*
* ENTRY REQUIREMENTS:   A - BYTE TO DISPLAY
*
* EXIT CONDITIONS:      ALL REGISTERS PRESERVED
*                       EXCEPT C
*
*********************************************************
OUTHEX  PSHS    A       SAVE THE BYTE
        LSRA
        LSRA
        LSRA
        LSRA
        BSR     OUTDIG  DISPLAY IT
        LDA     ,S      GET LS DIGIT
        BSR     OUTDIG  DISPLAY IT
        PULS    A,PC    RESTORE A & RETURN
*********************************************************
* DISPLAY A HEX DIGIT
*********************************************************
OUTDIG  ANDA    #$0F    MASK OFF DIGIT
        ADDA    #$30    CONVERT TO ASCII
        CMPA    #$39    BIGGER THAN 9?
        BLS     OUTCHR  GO IF NOT
        ADDA    #$07    CONVERT TO LETTER
        BRA     OUTCHR  PRINT & EXIT
*********************************************************
* PRINT A STRING TO THE CONSOLE
*
* ENTRY REQUIREMENTS:   X - POINTS TO STRING
*                       LAST BYTE HAS BIT 7 ON
*
* EXIT CONDITIONS:      X - POINTS 1 BYTE PAST END
*                       A,C - CHANGED
*
*********************************************************
PSTRNG  LDA     ,X      GET A CHARACTER
        ANDA    #$7F    MASK OFF
        BSR     OUTCHR  DISPLAY IT
        TST     ,X+     WAS IT LAST?
        BPL     PSTRNG  LOOP IF NOT
        RTS
*********************************************************
* PRINT CR/LF ON CONSOLE
*
* ENTRY REQUIREMENTS:   NONE
*
* EXIT CONDITIONS:      ALL REGISTERS PRESERVED
*                       EXCEPT C
*
*********************************************************
```

```
CRLF    PSHS    A       SAVE A REGISTER
        LDA     #CR     OUTPUT CR
        BSR     OUTCHR
        LDA     #LF     OUTPUT LF & EXIT
        BSR     OUTCHR
        BSR     NULL
        PULS    A,PC
*********************************************************
* SEND 4 NULLS TO CONSOLE
*
* ENTRY REQUIREMENTS:   NONE
*
* EXIT CONDITIONS:      ALL REGISTERS PRESERVED
*                       EXCEPT C
*********************************************************
NULL    PSHS    A,B
        CLRA
        LDB     #$4
NULL1   BSR     OUTCHR
        DECB
        BNE     NULL1
        PULS    A,B,PC
*********************************************************
* SAVE PROGRAM ON TAPE
*********************************************************
TSAVE   BSR     GETHX   GET START ADDRESS
        BEQ     TSAVE2  GO IF NONE
        STX     BEGADD  SAVE START
        BSR     GETHX   GET END ADDRESS
        BNE     TSAVE1  GO IF ENTERED
        LDX     BEGADD  DUPLICATE ADDRESS
        INCB            SET ADDRESS INDICATOR
TSAVE1  STX     ENDADD  SAVE END
TSAVE2  LDX     CODCB   SAVE CONSOLE DCB
        PSHS    A,X     SAVE TERMINATOR TOO
        LDX     TPDCB   SET UP FOR TAPE
        STX     CODCB
        TSTB            ANY ADDRESS ENTERED?
        BEQ     TSAVE3  GO IF NOT
        BSR     SAVE    SAVE THE PROGRAM
TSAVE3  PULS    A       GET TERMINATOR
        CMPA    #CR     WAS IT RETURN?
        BNE     TSAVE4  GO IF NOT
        LDB     #'9     OUTPUT S9 RECORD
        BSR     OUTSN
TSAVE4  PULS    X       RESTORE DCB POINTER
        STX     CODCB
        RTS
*********************************************************
* GET HEX NUMBER IN X
*********************************************************
GETHX   JMP     GETHEX  RELATIVE BRANCH BOOSTER
*********************************************************
* SAVE A PROGRAM IN HEX
*
* ENTRY REQUIREMENTS:   SAVE ADDRESSES ARE IN
*                       BEGADD & ENDADD
*
* EXIT CONDITIONS:      ALL REGISTERS CHANGED
*********************************************************
```

```
SAVE    LDX     BEGADD  POINT AT FIRST BYTE
SAVE1   LDB     #'1     BEGIN NEW S1 RECORD
        BSR     OUTSN
        CLR     CKSUM   INIT CHECKSUM
        LDD     ENDADD  CALCULATE BYTES TO SAVE
        PSHS    X
        SUBD    ,S++
        TSTA            GREATER THAN 255?
        BNE     SAVE2   GO IF YES
        CMPB    #16     LESS THAN FULL RECORD?
        BLO     SAVE3   GO IF YES
SAVE2   LDB     #15     SET FULL RECORD SIZE
SAVE3   INCB            CORRECT RECORD SIZE
        TFR     B,A     OUTPUT RECORD SIZE
        ADDA    #3      ADJUST FOR ADDRESS,COUNT
        BSR     OUTBYT
        PSHS    X       ADDRESS TO STACK
        PULS    A       OUTPUT ADDRESS HI
        BSR     OUTBYT
        PULS    A       OUTPUT ADDRESS LO
        BSR     OUTBYT
SAVE4   LDA     ,X+     SAVE A DATA BYTE
        BSR     OUTBYT
        DECB            LOOP UNTIL 0
        BNE     SAVE4
        LDA     CKSUM   GET CHECKSUM
        COMA            COMPLIMENT IT
        BSR     OUTBYT  OUTPUT IT
        LEAY    -1,X    CHECK FOR END
        CMPY    ENDADD
        BNE     SAVE1
        RTS
*********************************************************
* OUTPUT BYTE AS HEX AND ADD TO CHECKSUM
*********************************************************
OUTBYT  JSR     OUTHEX  OUTPUT BYTE AS HEX
        ADDA    CKSUM   ADD TO CHECKSUM
        STA     CKSUM
        RTS
*********************************************************
* OUTPUT 'S' TAPE RECORD HEADERS
*********************************************************
OUTSN   JSR     CRLF    BEGIN NEW LINE
        LDA     #'S     OUTPUT 'S' HEADER
        BSR OUTC
        TFR     B,A     RECORD TYPE TO A
*********************************************************
* OUTPUT CHARACTER TO CONSOLE
*********************************************************
OUTC    JMP     OUTCHR  RELATIVE BRANCH BOOST
*********************************************************
* MEMORY EXAMINE AND CHANGE
*********************************************************
MEMEC   BSR     GETHX   GET ADDRESS
        BNE     MEMEC1  GO IF GOOD
        LDX     MEMPTR  USE PREVIOUS
MEMEC1  STX     MEMPTR  UPDATE RAM POINTER
        JSR     CRLF    BEGIN NEW LINE
        TFR     X,D     DISPLAY ADDRESS
        JSR     DSPDBY
```

```
         LDA     ,X+     GET CONTENTS
         JSR     DSPSBY  DISPLAY THEM
         TFR     X,Y     SAVE ADDRESS IN Y
         BSR     GETHX   GET CHANGE DATA
         EXG     D,X     SAVE DELIM, GET NEW
         BEQ     MEMEC2  GO IF NO CHANGE
         STB     -1,Y    UPDATE MEMORY
         CMPB    -1,Y    VERIFY GOOD STORE
         BEQ     MEMEC2  GO IF GOOD STORE
         JSR     ERROR   DISPLAY ERROR
MEMEC2   TFR     X,D     GET DELIMITER IN A
         TFR     Y,X     GET NEXT ADDRESS IN X
         CMPA    #CR     END OF UPDATE?
         BEQ     MEMEC3  GO IF YES
         CMPA    #'^     BACKING UP?
         BNE     MEMEC1  LOOP IF NOT
         LEAX    ,--X    BACK UP 2
         BRA     MEMEC1  CONTINUE
MEMEC3   RTS
*********************************************************
* GO TO ADDRESS
*********************************************************
GO       LDS     STKPTR  SET UP STACK
         JSR     GETHEX  GET TARGET ADDRESS
         BEQ     GO1     GO IF NONE
         STX     10,S    STORE IN PC ON STACK
GO1      LDA     ,S      SET 'E' FLAG IN CC
         ORA     #$80
         STA     ,S
INTRET   RTI             LOAD REGISTERS AND GO
*********************************************************
* BREAKPOINT    (SOFTWARE INTERRUPT) TRAP
*********************************************************
BRKPNT   LDX     10,S    GET     PROGRAM COUNTER
         LEAX    -1,X    DECREMENT BY 1
         STX     10,S    REPLACE ON STACK
         LDB     #$FF    FLAG FOR SINGLE REMOVAL
         JSR     REMBK   REMOVE BREAKPOINT
*********************************************************
* INTERRUPT (HARDWARE/SOFTWARE) TRAP
*********************************************************
TRAP     STS     STKPTR  SAVE STACK POINTER
         JSR     CRLF    BEGIN NEW LINE
         BSR     REGDMP  DUMP REGISTERS
         JMP     GETCMD  GET NEXT COMMAND
*********************************************************
* REGISTER EXAMINE AND CHANGE
*********************************************************
REGEC    JSR     INCHR   GET REGISTER TO EXAMINE
         JSR     CRLF    BEGIN NEW LINE
         CLRB            CLEAR OFFSET COUNT
         LDX     #REGIDS POINT TO REGISTER ID STRING
REGEC1   CMPA    B,X     CHECK REGISTER NAME
         BEQ     REGEC2  GO IF FOUND
         INCB            ADVANCE COUNTER
         CMPB    #11     END OF LIST?
         BLS     REGEC1  LOOP IF NOT
         BRA     REGDMP  BAD ID - DUMP ALL
REGEC2   PSHS    B       SAVE OFFSET
         BSR     RDUMP   DISPLAY REG & CONTENTS
```

```
         JSR     GETHEX  GET NEW VALUE
         PULS    B       RESTORE OFFSET
         BEQ     REGECX  GO IF NO CHANGE
         LEAY    B,Y     POINT TO REG ON STACK
         CMPB    #3      SINGLE BYTE REG?
         TFR     X,D     GET NEW DATA IN A,B
         BLS     REGEC3  GO IF SINGLE
         STA     ,Y+     STORE MS BYTE
REGEC3   STB     ,Y      STORE LS BYTE
REGECX   RTS
REGIDS   FCC     'CABDXXYYUUPP'
*********************************************************
* COMPLETE REGISTER DUMP
*********************************************************
REGDMP   LDX     #REGIDS POINT TO ID STRING
         CLRB             CLEAR OFFSET COUNTER
RGDMP1   LDA     B,X     GET REG NAME
         BSR     RDUMP   DISPLAY IT
         INCB             DISPLAY IT
         CMPB    #11     ALL PRINTED?
         BLS     RGDMP1  LOOP IF NOT
         LDA     #'S     DISPLAY STACK ID
         BSR     DSPID
         LDY     #STKPTR-12  Y+B=>STKPTR
         BRA     RDUMP1
*********************************************************
* DISPLAY REGISTER CONTENTS
*********************************************************
RDUMP    BSR     DSPID   DISPLAY REGISTER ID
         LDY     STKPTR  POINT Y AT STACK
         CMPB    #3      SINGLE BYTE REG?
         BLS     RDUMP2  GO IF YES
RDUMP1   LDA     B,Y     DISPLAY MS BYTE
         JSR     OUTHEX
         INCB             ADVANCE OFFSET
RDUMP2   LDA     B,Y     DISPLAY A BYTE
         JMP     DSPSBY
*********************************************************
* DISPLAY REGISTER ID
*********************************************************
DSPID    BSR     OUTCH   DISPLAY REG NAME
         LDA     #'=     DISPLAY '='
*********************************************************
* OUTPUT CHARACTER TO CONSOLE
*********************************************************
OUTCH    JMP     OUTCHR  RELATIVE BRANCH BOOSTER
*********************************************************
* SET A BREAKPOINT
*********************************************************
SETBK    JSR     GETHEX  GET ADDRESS
         BEQ     DSPBK   GO IF NONE ENTERED
         BSR     INITBP  POINT Y AT BP TABLE
SETBK1   LDD     ,Y      EMPTY SLOT?
         BEQ     SETBK2  GO IF YES
         BSR     NEXTBP  ADVANCE TO NEXT SLOT
         BNE     SETBK1  LOOP IF NOT END
         BRA     DSPBK   EXIT
SETBK2   STX     ,Y      SAVE ADDRESS
         BEQ     DSPBK   GO IF ADDRESS = 0
         LDA     ,X      GET CONTENTS
```

```
        STA     2,Y     SAVE IN TABLE
        LDA     #$3F    SWI OP CODE
        STA     ,X      SET BREAK
*********************************************************
* DISPLAY ALL BREAKPOINTS
*********************************************************
DSPBK   JSR     CRLF    BEGIN NEW LINE
        BSR     INITBP  POINT Y AT BP TABLE
DSPBK1  LDD     ,Y      GET ADDRESS OF BP
        BEQ     DSPBK2  GO IF INACTIVE
        JSR     DSPDBY  DISPLAY ADDRESS
DSPBK2  BSR     NEXTBP  ADVANCE POINTER
        BNE     DSPBK1  LOOP IF NOT END
        RTS
*********************************************************
* INITIALISE BREAKPOINT TABLE POINTER
*********************************************************
INITBP  LDY     #BPTABL POINT Y AT BP TABLE
        RTS
*********************************************************
* ADVANCE BREAKPOINT TABLE POINTER
*********************************************************
NEXTBP  LEAY    3,Y     ADVANCE TO NEXT ENTRY
        CMPY    #BPTEND CHECK FOR END OF TABLE
        RTS
*********************************************************
* UNSET A BREAKPOINT
*********************************************************
UNSBK   JSR     GETHEX  GET ADDRESS
*********************************************************
* REMOVE ONE OR MORE BREAKPOINTS
*********************************************************
REMBK   BSR     INITBP  POINT Y AT BP TABLE
REMBK1  TSTB            REMOVE ALL?
        BEQ     REMBK2  GO IF YES
        CMPX    ,Y      FIND ADDRESS?
        BEQ     UNSET   GO IF YES
        BRA     REMBK3  LOOP IF NO
REMBK2  BSR     UNSET   UNSET IT
REMBK3  BSR     NEXTBP  ADVANCE POINTER
        BNE     REMBK1  LOOP IF NOT END
        RTS
*********************************************************
* REMOVE A BREAKPOINT
*********************************************************
UNSET   LDX     ,Y      GET ADDRESS OF BP
        BEQ     UNSET1  GO IF INACTIVE
        LDA     2,Y     GET CONTENTS
        STA     ,X      REPLACE BP
        CLR     0,Y     MARK BP INACTIVE
        CLR     1,Y
UNSET1  RTS
*********************************************************
* TRANSPARENT LINK BETWEEN CONSOLE AND REMOTE
*********************************************************
TRANSP  LDX     #CONDCB   SELECT CONSOLE DEVICE
        LDB     #STATFN
        JSR     REQIO     GET ITS STATUS
        BITA    #1        ANY CHARS WAITING ?
        BEQ     TRANS2    BRANCH IF NO
```

```
         LDB     #READFN
         JSR     REQIO       READ CHARACTER
         CMPA    #$1B        IS IT <ESC> ?
         BNE     TRANS1      BRANCH IF NO
         RTS                 EXIT TRANSPARENT MODE
TRANS1   LDX     #REMDCB     SELECT REMOTE DEVICE
         LDB     #WRITFN
         JSR     REQIO       SEND CHARACTER
TRANS2   LDX     #REMDCB     SELECT REMOTE DEVICE
         LDB     #STATFN
         JSR     REQIO       GET ITS STATUS
         BITA    #1          ANY CHARS WAITING ?
         BEQ     TRANSP      BRANCH IF NO
         LDB     #READFN
         JSR     REQIO       READ CHARACTER
         LDX     #CONDCB     SELECT CONSOLE DEVICE
         LDB     #WRITFN
         JSR     REQIO       SEND CHARACTER
         BRA     TRANSP      LOOP FOR NEXT CHAR
*************************************************************
* TERMINAL DRIVER (ACIA)
*************************************************************
TERMDR   CLR     DCBERR,X          NO ERRORS POSSIBLE
         LDX     DCBIOA,X          GET I/O ADDRESS
         LSRB                      READ FUNCTION?
         BCS     TERMRD            GO IF YES
         LSRB                      WRITE FUNCTION?
         BCS     TERMWT            GO IF YES
         LSRB                      STATUS FUNCTION?
         BCS     TERMST            GO IF YES
         LSRB                      CONTROL FUNCTION?
         BCC     TERM1             GO IF NOT
         STA     ,X                STORE CONTROL CODE
TERM1    RTS
TERMRD   LDB     ,X          GET STATUS
         LSRB                 INPUT BIT TO C
         BCC     TERMRD      LOOP IF NO INPUT
         LDA     1,X         GET CHARACTER
         RTS
TERMWT   LDB     ,X          GET STATUS
         BITB    #2          READY FOR OUTPUT?
         BEQ     TERMWT      LOOP IF NOT
         STA     1,X         OUTPUT CHARACTER
         RTS
TERMST   LDA     ,X          GET STATUS
         ANDA    #3          MASK OFF READY BITS
         RTS
*************************************************************
* PR9MON COMMAND TABLE
*************************************************************
CMDTBL   FCB     1           ITEM LENGTH
         FCB     'M          MEMORY EXAMINE/CHANGE
         FDB     MEMEC
         FCB     'G          GO TO ADDRESS
         FDB     GO
         FCB     'L          PROGRAM LOAD
         FDB     TLOAD
         FCB     'S          PROGRAM SAVE
         FDB     TSAVE
         FCB     'R          REGISTER EXAMINE/CHANGE
```

```
        FDB     REGEC
        FCB     'B      SET/PRINT BREAKPOINTS
        FDB     SETBK
        FCB     'U      UNSET BREAKPOINTS
        FDB     UNSBK
        FCB     'T      ENTER TRANSPARENT MODE
        FDB     TRANSP
        FCB     'Z      JUMP TO EPROM AT $3000
        FDB     $3000
        FCB     $FF
*********************************************************
* RAM INITIALISATION DATA
*********************************************************
RAMINT  FCC     'CN'    CONSOLE DCB ID
        FDB     TERMDR  CONSOLE DRIVER
        FDB     TERMNL  CONSOLE I/O ADDRESS
        FDB     0       ERROR STATUS, EXT
        FDB     0       REMOTE DCB
        FCC     'RM'    ID
        FDB     TERMDR  DRIVER
        FDB     REMOTE  I/O ADDRESS
        FDB     0       ERR STAT/EXT
        FDB     CONDCB  DCB CHAIN POINTER
        FDB     CONDCB  DCB POINTERS
        FDB     CONDCB
        FDB     CONDCB
        FDB     REMDCB
        FDB     TRAP    INTERRUPT VECTORS
        FDB     TRAP
        FDB     INTRET
        FDB     TRAP
        FDB     BRKPNT
        FDB     TRAP
        END
```

A.5.2 COLSYS OPERATING PROGRAM

```
         NAM      COLSYS
         OPT      NOG,LLE=120,P=62
*
*
*        PROGRAM TO OPERATE 9" DISTILLATION COLUMN
*        =========================================
*
*                 P.W.M.Janssen
*
*                 24th March 1986
*
*
* NB. This program must be preassembled to change values
*     inside brackets to AM9511 floating point format.
*
*************************************************************
* PR9MON Monitor callable subroutines
*************************************************************
*
OUTCHR   EQU      $2190    Output character to console
DSPSBY   EQU      $21AB    Display single byte & space
DSPDBY   EQU      $21A2    Display double byte & space
OUTHEX   EQU      $21B5    Display 2 hex digits
PSTRNG   EQU      $21CF    Display string on console
CRLF     EQU      $21DA    Begin new line on console
OUTSP    EQU      $21AD    Output space to console
*
*************************************************************
* AM9511 callable subroutines
*************************************************************
*
PUSH16   EQU      $60EB    Push 16 bit number on stack
PUSH32   EQU      $60DD    Push 32 bit number on stack
PULL16   EQU      $6111    Pull 16 bit number from stack
PULL32   EQU      $60F9    Pull 32 bit number from stack
FLTS     EQU      $608C    Float single precision number
FLTD     EQU      $6091    Float double precision number
FIXS     EQU      $6082    Fix to single precision number
FIXD     EQU      $6087    Fix to double precision number
PTOF     EQU      $60AF    Duplicate floating point number
SADD     EQU      $6000    Single precision addition
SSUB     EQU      $6005    Single precision subtraction
SDIV     EQU      $6014    Single precision division
DADD     EQU      $6019    Double precision addition
DDIV     EQU      $602D    Double precision division
FADD     EQU      $6032    Floating point addition
FSUB     EQU      $6037    Floating point subtraction
FMUL     EQU      $603C    Floating point multiplication
FDIV     EQU      $6041    Floating point division
*
```

```
*********************************************************
* System hardware addresses
*********************************************************
*
* ACIA control/status registers
*
TERMNL  EQU     $FBCE           Terminal ACIA
COMPUT  EQU     $DFB8           Computer ACIA
*
* Data aquisition module
*
DMSB    EQU     $FBC8
DCSR1   EQU     DMSB+1
DLSB    EQU     DCSR1+1
DCSR2   EQU     DLSB+1
*
* D/A PIAs
*
P5      EQU     $DF90           Steam
P4      EQU     $FBC2           Bottoms
P3      EQU     P4-2            Distillate
P2      EQU     P4+4            Reflux
P1      EQU     P4+2            Feed
P5CSR   EQU     P5+1            Control/status registers
P4CSR   EQU     P4+1
P3CSR   EQU     P3+1
P2CSR   EQU     P2+1
P1CSR   EQU     P1+1
*
* Solenoid valves PIA
*
SOLV    EQU     $DF92           Data register
SOLVCS  EQU     SOLV+1          Control/status register
*
* Programmable timer modules
*
TCS1_3  EQU     $DFA0           Control/status registers
TCSR2   EQU     $DFA1
TCS4_6  EQU     $DFA8
TCSR5   EQU     $DFA9
TCS7_9  EQU     $DFB0
TCSR8   EQU     $DFB1
COUNT1  EQU     TCSR2+1         Counter addresses
COUNT2  EQU     COUNT1+2
COUNT3  EQU     COUNT2+2
COUNT4  EQU     TCSR5+1
COUNT5  EQU     COUNT4+2
COUNT6  EQU     COUNT5+2
COUNT7  EQU     TCSR8+1
COUNT8  EQU     COUNT7+2
COUNT9  EQU     COUNT8+2
*
* Interupt & reset vectors
*
SWI3V   EQU     $FFF2           Software interupt 3
SWI2V   EQU     $FFF4           Software interupt 2
FIRQV   EQU     $FFF6           Fast interupt request
IRQV    EQU     $FFF8           Interupt request
SWIV    EQU     $FFFA           Software interupt
NMIV    EQU     $FFFC           Non-maskable interupt
```

```
RESTRT  EQU     $FFFE           Re-entry into PR9MON
*
*********************************************************
* Definition of program constants
*********************************************************
*
* Controller offsets
*
PROVAR  EQU     0               Address of process variable
CONVAR  EQU     2               Address of control variable PIA port
SLEWDN  EQU     4               Downward slew rate limit
SLEWUP  EQU     6               Upward slew rate limit
SETPNT  EQU     8               Set point (16 bit fixed point)
COEF1   EQU     10              K1 (32 bit floating point)
COEF2   EQU     14              K2 (32 bit floating point)
ERPRES  EQU     18              Present error (16 bit fixed point)
ERPREV  EQU     20              Previous error (16 bit fixed point)
CONOUT  EQU     22              Controller output (32 bit fixed)
*
* Serial I/O handler offsets
*
SOURCE  EQU     0                 Source ACIA address
DESTIN  EQU     2                 Destination ACIA address
RINGBF  EQU     4                 Ring buffer start
RECPTR  EQU     132               Last char received pointer
ACTPTR  EQU     133               Last char actioned pointer
INSBUF  EQU     134               Instruction buffer start
INSPTR  EQU     253               INSBUF next char pointer
INSFLG  EQU     255               Instruction receiving flag
*
* Digital filtering offsets
*
FILOUT  EQU     0               Filter output (16 bit fixed) address
MEASUR  EQU     2               Unfiltered measurement address
FILT32  EQU     4               Filter output (32 bit float) address
TIMCON  EQU     6               Filter time constant in seconds
*                                           (32 bit float point)
*
* Character definitions
*
INSCHR  EQU     '#                Instruction starting character
PASCHR  EQU     $10               Pass all toggle character
*
*********************************************************
* Ram data base
*********************************************************
        ORG     $0
*
* Raw plant data
*
SYSTAT  RMB     1                 System status
DAMERR  RMB     1                 D.A.M. error flag
DAM     RMB     32                D.A.M. data area
FLOW    RMB     8                 Turbine meter data
CELL1   RMB     34                Capacitance cell frequencies
CELL2   RMB     34
CELL3   RMB     34
CELLR   RMB     34
REFRAC  RMB     8                 Refrac data area
CSTAT   RMB     1
```

```
*
* Processed plant data
*
PRODAT  RMB     12              Filtered data ready for output
F32DAT  RMB     24              Filtered intermediate data
*
* Controller data area
*
CNTLR1  RMB     CONOUT+4        Feed via flowmeter
CNTLR2  RMB     CONOUT+4        Reflux via flowmeter
CNTLR3  RMB     CONOUT+4        Reflux drum level
CNTLR4  RMB     CONOUT+4        Reboiler level
PMPTMP  RMB     2               Temporary storage
*
* Watch dog timers
*
TIM1    RMB     1               Auto-update report timer
TIM2    RMB     1               Refractometer timer
TIM3    RMB     1               Feed mixing control timer
TIM4    RMB     1                "     "       "      "
TIM5    RMB     1               Alarm timer
*
* Refractometer paramters
*
RSTAT   RMB     1               Refrac present channel
RESEQ   RMB     12              Sampling sequence
SEQPTR  RMB     1               Sequence pointer
TRMIN   RMB     1               Sampling length
*
* Serial I/O handler RAM
*
TERBUF  RMB     256     Terminal input buffers area
COMBUF  RMB     256     Computer input buffers area
BUFFUL  RMB     1       Computer buffer full flag
TEMPTR  RMB     2       Temporary pointer storage
*
* User request handler RAM
*
BEGADD  RMB     2       Dump routine begining address
BYTNUM  RMB     2       Dump routine byte number
REPTIM  RMB     1       Auto-report routine timer
AUTOFG  RMB     1       Auto-report flag
AUACIA  RMB     2       Auto-report destination ACIA
SRFLAG  RMB     1       Report flag
SRACIA  RMB     2       Report destination ACIA
DECDIG  RMB     5       Dec -> binary conv storage
SNDASC  RMB     5       Bin -> 5 ASCII dec storage
DECHAR  RMB     1        "   "   "   "   "   "
FORMAT  RMB     1       0 => dec , else => hex
ECHOFL  RMB     1       0 => echo , else => no echo
PASSAL  RMB     1       0 => intercept # , else => passal
LFFLAG  RMB     1       Line-feed received flag
ALARMF  RMB     1       Alarm checking, 0 => enable
ALARMC  RMB     1       Alarm condition, 0 => no, else => yes
FMODE   RMB     1       Feed mixing mode
MRATIO  RMB     1       Mixng routine
FMTIM   RMB     1       Slow software timer for feed mixing
COMPBF  RMB     2       Storage area for mixing routine
```

```
*
* User and hardware stack area
*
        ORG     $FD00   Top of user stack
USRSTK  EQU     *
        ORG     $FF90   Top of hardware stack
HRDSTK  EQU     *
*********************************************************
* Definition of I/O channels
*********************************************************
*
* Data aquisition module channel specifications
*
TEMP1   EQU     DAM+2           Plate 1 temperature
TEMP5   EQU     DAM+8           Feed plate temperature
TEMP8   EQU     DAM+12          Plate 8 temperature
TCELL1  EQU     DAM+14          Cell 1 temperature
TCELL2  EQU     DAM+16          Cell 2 temperature
TCELL3  EQU     DAM             Cell 3 temperature
REFLVL  EQU     DAM+20          Reflux level
REBLVL  EQU     DAM+22          Reboiler level
*
* Turbine flow meter specifications
*
REFFLO  EQU     FLOW            Reflux
TOPFLO  EQU     FLOW+2          Tops
BOTFLO  EQU     FLOW+4          Bottoms
FEDFLO  EQU     FLOW+6          Feed
*********************************************************
* Main program
*********************************************************
        ORG     $3000
*
* System initialisation
*
START   LDS     #HRDSTK         Set up stack pointers
        LDU     #USRSTK
*
        LDX     #$FFF           Zero 4k of RAM
INITL1  CLR     ,X
        LEAX    -1,X
        BNE     INITL1
*
        LDX     #RAMINT         Initialise user RAM
INITL2  LDA     ,X+             Get byte count
        BEQ     INITL4          Branch if end of table
        LDY     ,X++            Get destination address
INITL3  LDB     ,X+             Get data byte
        STB     ,Y+             Store it at destination
        DECA                    Decrement byte count
        BEQ     INITL2          Branch if more bytes
        BRA     INITL3          Branch for next data set
*
* PIA Initialisation
*
INITL4  CLRA
        STA     DCSR1           Point to data direction regs
        STA     DCSR2
        STA     P1CSR
        STA     P2CSR
```

```
        STA     P3CSR
        STA     P4CSR
        STA     P5CSR
        STA     SOLVCS
*
        STA     DMSB            Set PIAS for input
        STA     DLSB
        COMA
        STA     P1              Set PIAS for output
        STA     P2
        STA     P3
        STA     P4
        STA     P5
        STA     SOLV
*
        LDA     #$36
        STA     DCSR1           Set up control registers
        STA     DCSR2
        STA     P1CSR
        STA     P2CSR
        STA     P3CSR
        STA     P4CSR
        STA     P5CSR
        STA     SOLVCS
*
        LDD     #997            Init 1 sec intervals
        STD     COUNT5
        LDA     #1
        STA     TCSR2
        STA     TCSR5
        STA     TCSR8
        CLR     TCS1_3          Set counters going
        CLR     TCS4_6
        CLR     TCS7_9
*
        LDD     #FIRSER         Set fast int vector
        STD     FIRQV
        LDD     #IRQSER         Set IRQ int vector
        STD     IRQV
        LDD     #START          Set NMI int vector
        STD     NMIV
        LDA     #$03            Configure both ACIAS &
        STA     TERMNL          enable ACIA interrupts
        STA     COMPUT
        LDA     #$01
        STA     TERMNL
        STA     COMPUT
        LDA     TERMNL+1
        LDA     COMPUT+1
        LDA     #$81
        STA     TERMNL
        STA     COMPUT
        LDA     #$40            Enable PTM interrupts
        STA     TCSR5
        ANDCC   #$AF            Clear all int masks
*
* Print welcome and identification message on terminal
*
        JSR     CRLF            New line
        LDX     #WELMES         Point to message
```

```
        JSR     PSTRNG          Print it
        JSR     CRLF            New line
        JSR     MAIN            Start main loop
*
WELMES  FCC     'Distillation Column Operating Software '
        FCC     'Version 2.0'
        FCB     $80
*
* Low priority serial I/O handler
*
MAIN    LDX     #TERBUF         Point at terminal buffer
        LDB     RECPTR,X
        SUBB    ACTPTR,X
        BEQ     MAIN4           Branch if no data
        LDA     ACTPTR,X        Increment data
        JSR     INCPTR          Actioned pointer
        STA     ACTPTR,X
        LDA     A,Y             Get character
        CMPA    #PASCHR         Is it <CNTRL>/P ?
        LBEQ    CNTRLP
        LDB     INSFLG,X        Part of instruction to micro
        LBNE    INSTRT          Branch if yes
        LDB     PASSAL          In passal mode
        BNE     MAIN9           Branch if yes
        CMPA    #INSCHR         Is it hash <#> ?
        LBEQ    MAIN2
MAIN9   CMPA    #$20            Is it printable ?
        BGE     MAIN8           Go if yes
        CMPA    #$13            Is it <CNTRL>/S ?
        BNE     MAIN1
        LDB     BUFFUL          If yes, set bit 1 of
        ORB     #$2             BUFFUL flag
        STB     BUFFUL
MAIN1   CMPA    #$11            Is it <CNTRL>/Q ?
        BNE     MAIN3
        CLR     BUFFUL          If yes, clear BUFFUL flag
MAIN3   CMPA    #$19            Is it <CNTRL>/Y ?
        BEQ     MAIN7
        CMPA    #$03            Is it <CNTRL>/C ?
        BNE     MAIN8
MAIN7   CLR     AUTOFG          Stop any current auto-reports
MAIN8   LEAX    [DESTIN,X]      Point at destination ACIA
        JSR     SEND            Send character
*
MAIN4   LDX     #COMBUF         Point at computer buffer
        LDB     RECPTR,X
        SUBB    ACTPTR,X
        BEQ     MAIN            Branch if no data
        BGT     MAIN5
        ANDB    #$7F            B reg holds no of characters
MAIN5   LDA     BUFFUL
        BITA    #$1             Has <CNTRL>/S been sent ?
        BEQ     MAIN6           Branch if not
        CMPB    #16             Number of chars > 16 ?
        BGT     MAIN6           Branch if yes
        ANDA    #$FE            Clear <CNTRL>/S sent flag
        STA     BUFFUL
        BITA    #$2             No scroll from terminal ?
        BNE     MAIN6           Branch if yes
        LDX     #COMPUT         Send <CNTRL>/Q if less than
```

```
         LDA     #$11            16 characters in buffer and
         JSR     SEND            <CNTRL>/S previously sent
         LDX     #COMBUF
MAIN6    LDA     ACTPTR,X        Increment actioned pointer
         JSR     INCPTR
         STA     ACTPTR,X
         LDA     A,Y             Get character
         LDB     INSFLG,X        Part of instruction to micro
         BNE     INSTRT          Branch if yes
         LDB     PASSAL          In passal mode
         BNE     MAIN10          Branch if yes
         CMPA    #INSCHR         Is it hash "#" ?
         BEQ     MAIN2
MAIN10   CMPA    #$0A            Is it <LF> ?
         BNE     MAIN12          Branch if no
         LDB     LFFLAG          Extract <LF> ?
         BEQ     MAIN12          Branch if no
         CLR     LFFLAG          Clear extract <LF> flag
         JMP     MAIN            and ignore <LF> char
MAIN12   LEAX    [DESTIN,X]
         JSR     SEND            Send character
         JMP     MAIN
*
MAIN2    STA     INSFLG,X        Set insruction receiving flag
         CMPX    #TERBUF         Is instruction from terminal ?
         LBNE    MAIN            Branch if not
         LDX     #INSMES
         JSR     PSTRNG          Print message
         JMP     MAIN
*
INSMES   FCB     $0D,$0A,0,0,0,0
         FCC     'INS'
         FCB     ' +$80
*
* Put character into instruction buffer and test for terminater
*                terminater:- <CR> for terminal
*                             <LF> for computer
*
INSTRT   CMPX    #TERBUF         Is char from terminal ?
         BNE     INSTR4          Branch if not
         CMPA    #$7F            Is it delete char ?
         BNE     INSTR4          Branch if not
         LDY     INSPTR,X        Fetch pointer
         CMPY    #TERBUF+INSBUF  Is it start of buffer ?
         LBEQ    MAIN            Return if yes
         LEAY    -1,Y            Decrement pointer
         STY     INSPTR,X        and save
         LDX     #TERMNL
         LDA     #$08            Send backspace,blank,-
         JSR     SEND            backspace
         LDA     #$20
         JSR     SEND
         LDA     #$08
         JSR     SEND
         JMP     MAIN            Return
INSTR4   LDY     INSPTR,X
         STA     ,Y+             Store char in instruction buffer
         STY     INSPTR,X
         CMPX    #COMBUF         Which buffer ?
         BEQ     INSTR2
```

```
         CMPA    #$0D            Is it <CR> terminater
         BEQ     INSTR1          Branch if yes
         LDX     #TERMNL
         JSR     SEND            Print char on terminal
         JMP     MAIN
INSTR2   CMPA    #$0D            Is it <CR> terminater
         BEQ     INSTR7           Branch if yes
         CMPY    #COMBUF+INSBUF+2 Is it second char ","
         LBNE    MAIN             Branch if not
         LDA     -2,Y            Get first char in buffer
         CMPA    #'I             Is char i from report
         LBNE    MAIN            Branch if not
         LDA     ECHOFL          Echo on or off ?
         LBNE    MAIN            Branch if off
         CLR     INSFLG,X
         LEAY    INSBUF,X
         STY     INSPTR,X
         JMP     MAIN
INSTR7   LDA     #$FF            Set <LF> extract flag
         STA     LFFLAG
INSTR1   CLR     INSFLG,X        Clear receiving instruction
         LEAY    INSBUF,X        flag
         STY     INSPTR,X
         JSR     ACTION          Action instruction
         JMP     MAIN
*
* <CNTRL>/P character recieved
*
CNTRLP   CLR     AUTOFG          Stop any current auto reports
         LDA     PASSAL          In pass all mode ?
         BEQ     CNTP1           Branch if no
         CLR     PASSAL          Set intercept mode
         LDA     #$34            Turn LED off
         BRA     CNTP2
CNTP1    LDA     #$3C            Turn LED on
         STA     PASSAL          Set pass all mode
CNTP2    STA     DCSR1
         JMP     MAIN
*
* Increment ring buffer pointer
*
INCPTR   INCA                    Add one to pointer
         ANDA    #$7F            Loop it around if needed
         LEAY    RINGBF,X        Point at ring buffer
         RTS
*
* Send character
*       Entry: X - ACIA C/S register address
*              A - ASCII encoded character
*
SEND     PSHS    A               Save character
SEND1    LDA     ,X              Get ACIA status
         BITA    #$2             Ready to send ?
         BEQ     SEND1           Loop if not
         PULS    A               Restore character
         STA     1,X             Send it
         RTS
```

```
*********************************************************
* Fast interrupt serial I/O handler
*********************************************************
FIRSER  PSHS    A,B,X,Y         Save registers
        LDX     #COMBUF         Point at computer ACIA data
        LDA     [SOURCE,X]      Get computer ACIA status
        LSRA                    Is it interrupting ?
        BCS     FIRSR1          Branch if yes
        LDX     #TERBUF         Point at terminal ACIA data
FIRSR1  LDY     ,X              Point at source ACIA
        LDB     1,Y             Get character
        ANDB    #$7F            Clear bit 7
        LDA     RECPTR,X
        JSR     INCPTR          Increment ring buffer pointer
        STA     RECPTR,X
        STB     A,Y             Store char in ring buffer
        SUBA    ACTPTR,X        Get number of characters
        BGT     FIRSR3
        ANDA    #$7F
FIRSR3  LDB     BUFFUL
        BITB    #$1             Has <CNTRL>/S been sent ?
        BNE     FIRSR4          Branch if yes
        CMPA    #96             More than 96 chars ?
        BLT     FIRSR4          Branch if no
        ORB     #$1             Set <CNTRL>/S sent flag
        STB     BUFFUL
        LDA     #$13
        LDX     #COMPUT
        JSR     SEND            Send <CNTRL>/S
FIRSR4  PULS    A,B,X,Y         Restore registers
        RTI
*********************************************************
* Action user request
*********************************************************
ACTION  LDA     INSBUF,X        Get first character
        JSR     UPCASE
        LDY     #CMDTBL         Point at table
        JSR     SEARCH          Scan table
        BMI     ACT1            Exit if no match
        LEAX    INSBUF+1,X
        JMP     [1,Y]           Jump to routine
ACT1    RTS
*
CMDTBL  FCB     'A
        FDB     AUTSET          Auto-report
        FCB     'C
        FDB     CONSET          Controller set points
        FCB     'D
        FDB     DUMP            Memory dump
        FCB     'E
        FDB     ECHOST          Echo switch
        FCB     'F
        FDB     MIXSET          Feed mixing
        FCB     'H
        FDB     FORMST          Hex/decimal format
        FCB     'P
        FDB     PSTAT           Pump status
        FCB     'R
        FDB     REPSET          Single report
        FCB     'S
```

```
        FDB     MEMSET          Memory set
        FCB     'T
        FDB     TANKS           Tank status
        FCB     'V
        FDB     VSTAT           Valve status
        FCB     'W
        FDB     WARN            Warning switch
        FCB     'X
        FDB     EXAMIN          Solenoid manifold
        FCB     'Z
        FDB     SHUTDN          Shut down procedure
        FCB     $FF
*
* Routine to search table for character
*       Entry: A - character
*              Y - points at start of table
*       Exit: Y  - points at matching character in table
*             CC - set according to -ve => no match
*
SEARCH  CMPA    ,Y              Compare to table entry
        BEQ     SEAR1           Branch if match
        LEAY    3,Y             Point to next item
        TST     ,Y              End of table ?
        BPL     SEARCH          Loop if not
        RTS
SEAR1   TST     ,Y              Set CC register
        RTS
*
* Routine to dump a specified number of memory bytes,
* previous values are used if not fully specified.
*       Form:- D <add> <bytes>
*
DUMP    JSR     CRLF            New line
        JSR     GETHEX          Get start address
        BEQ     DUMP2           Branch if unspecified
        STD     BEGADD          Save start address
        JSR     GETHEX          Get byte count
        BEQ     DUMP2           Branch if unspecified
        STD     BYTNUM          Save byte count
DUMP2   LDY     BYTNUM          Get byte count
        LDX     BEGADD          Get start address
DUMP3   EXG     D,X
        JSR     DSPDBY          Print address
        EXG     D,X
DUMP4   LDA     ,X+             Get memory contents
        JSR     DSPSBY          Print them
        LEAY    -1,Y            Decrement byte counter
        BEQ     DUMP5           Branch if done
        EXG     D,X
        BITB    #$0F            At 16 byte boundary ?
        EXG     D,X
        BNE     DUMP4           Branch if no
        JSR     CRLF            New line
        BRA     DUMP3           Loop for more
DUMP5   JSR     CRLF            New line
        RTS
```

```
*
* Routine to set specified memory locations,
* instruction ignored if incorrectly issued.
*       form:- M <add> <byte> <byte>....<byte>
*
MEMSET  JSR     GETHEX          Get address
        BEQ     MEMS2           Do nothing if unspecified
        EXG     D,Y             Shift address into Y reg
MEMS1   JSR     GETHEX          Get byte
        BEQ     MEMS2           Branch if finished
        STB     ,Y+             Set memory location
        BRA     MEMS1           Branch for next byte
MEMS2   RTS
*
* Routine to display or set solenoid valve control byte.
*       Forms: to display - V
*              to set - V <byte>
*
VSTAT   JSR     GETHEX          Get new valve position
        BEQ     VSTAT3          Branch if unspecified
        ANDB    #$3F            Clear insignificant bits
        STB     SOLV            Output to solenoid valves
        RTS
*
VSTAT3  JSR     CRLF            New lines
        JSR     CRLF
        LDX     #VSTMES         Point at message
        JSR     PSTRNG          Print it
        LDA     SOLV            Get valve position
        JSR     DSPSBY          Print it
        JSR     CRLF            New line
        RTS
VSTMES  FCC     'SOLENOID VALVES -'
        FCB     ' +$80
*
* Routine to select sample channel at solenoid manifold
*       Forms: to select - #X <Ch No>
*
*               channel numbers: 0 - distillate
*                                1 - bottoms
*                                2 - feed
*                                3 - reference
*
EXAMIN  JSR     GETHEX          Get channel number
        BEQ     RE10            Do nothing if unspecified
        ANDB    #$3             Clear insignificant bits
        ASRB                    Action bit 0 of byte
        BHS     RE6
        LDA     #$3E            Raise PIA control line
        BRA     RE7
RE6     LDA     #$36            Lower PIA control line
RE7     STA     P2CSR
        ASRB                    Action bit 1 of byte
        BHS     RE8
        LDA     #$3E            Raise PIA control line
        BRA     RE9
RE8     LDA     #$36            Lower PIA control line
RE9     STA     P1CSR
RE10    RTS
*
```

```
* Routine to display or change pump settings.
*       Forms: to display all - P
*              to display one - P <pump_number>
*              to change one  - P <pump_number> <byte>
*
PSTAT   JSR     CRLF            New line
        JSR     GETNUM          Get <pump_number>
        BNE     PSTAT1          Branch if specified
        LDA     #$1             Select pump 1
PSTAT6  PSHS    A
        JSR     PSTAT5          Display its status
        PULS    A
        INCA                    Select next pump
        CMPA    #$5
        BLE     PSTAT6          Loop if not all done
        RTS
*
PSTAT1  PSHS    B               Save <pump_number>
        JSR     GETNUM          Get <byte>
        BNE     PSTAT2          Branch if speciifed
        PULS    A               Get <pump_number>
PSTAT5  LDY     #PMPTAB         Point at address table
        JSR     SEARCH          Scan table
        BMI     PSTAT4          Exit if no match
        LDX     #PMPMES
        JSR     PSTRNG          Print "PUMP" string
        LDA     ,Y
        JSR     DSPSBY          Print <pump_number>
        CLRA                    Get pump setting
        LDB     [1,Y]
        LDX     #TERMNL
        JSR     SNDNUM          Send setting
        JSR     CRLF            New line
        RTS
*
PSTAT2  PULS    A               Get <pump_number>
        LDY     #PMPTAB         Point at address table
        JSR     SEARCH          Scan table
        BMI     PSTAT4          Exit if no match
        STB     [1,Y]           Set pump position
PSTAT4  RTS
*
PMPTAB  FCB     1               Pump address table
        FDB     P1
        FCB     2
        FDB     P2
        FCB     3
        FDB     P3
        FCB     4
        FDB     P4
        FCB     5
        FDB     P5
        FCB     $FF             End of table marker
*
PMPMES  FCC     'PUMP'
        FCB     $A0
*
* Routine to report tank levels
*
TANKS   JSR     CRLF            New line
```

```
        LDX     #TNKMES         Point at message table
        JSR     PSTRNG          Print first message
        LDD     DAM+28          Get feed tank 0 level
        JSR     TANK1           Preprocess measurement
        LDY     #$2800          Point at lookup table
        LDA     B,Y             Convert level -> volume%
        JSR     OUTHEX          Output volume
        JSR     CRLF            New line
        JSR     PSTRNG          Print second message
        LDD     DAM+30          Get feed tank 1 level
        JSR     TANK1           Preprocess measurement
        LDA     B,Y             Convert level -> volume%
        JSR     OUTHEX          Output volume
        JSR     CRLF            New line
        JSR     PSTRNG          Print third message
        LDD     DAM+24          Get distillate tank level
        JSR     TANK1           Preprocess measurement
        LDA     #141
        MUL                     Scale 0 to 99%
        EXG     A,B
        JSR     BINBCD          Convert to BCD
        JSR     OUTHEX          Output volume
        JSR     CRLF            New line
        JSR     PSTRNG          Print last message
        LDD     DAM+26          Get bottoms tank level
        JSR     TANK1           Preprocess measurement
        LDA     #115
        MUL                     Scale 0 to 99%
        EXG     A,B
        JSR     BINBCD          Convert to BCD
        JSR     OUTHEX          Output volume
        JSR     CRLF            New line
        RTS
*
TANK1   SUBD    #$100           Remove DP sensor offset
        BGE     TANK2           Is level +ve ?
        LDD     #$0
TANK2   LSRA                    Divide by 16
        RORB
        LSRA
        RORB
        LSRA
        RORB
        LSRA
        RORB
        RTS
*
TNKMES  FCC     'FEED 0  -'
        FCB     $A0
        FCC     'FEED 1  -'
        FCB     $A0
        FCC     'TOPS    -'
        FCB     $A0
        FCC     'BOTTOMS -'
        FCB     $A0
*
* Routine to convert binary number to BCD
*
*       Entry:   B - binary number 0-63
*       Exit:    A - BCD number 0-99
```

```
*
CONB7   FCB     $64,$32,$16     Table of constants
*
BINBCD  CLRA                    Clear BCD accumulator
        PSHS    B               Save binary number
        ANDB    #$F             Only retain LS nibble
        PSHS    B               Save it
        ADDA    ,S+             Add LS nibble to BCD
        DAA                     accumulator
*
        LDY     #CONB7-1        Point at constants
        PULS    B               Restore binary number
        ANDB    #$F0            Only retain MS nibble
        ASLB                    Ignore MS bit
*
BITSR   LEAY    1,Y             Point to next constant
        ASLB                    Shift bit to carry
        BCC     NEXTB           Branch if bit is zero
        ADDA    0,Y             Add constant from -
        DAA                     Table in BCD arithmetic
NEXTB   TSTB
        BNE     BITSR           Any bits left in B ?
        RTS
*
* Routine to initiate control of feed composition
*
MIXSET  JSR     GETNUM          Get mixing mode
        CMPB    #1              Mode 1 ?
        BNE     MIXSE1          Branch if no
        STB     FMODE           Save mode number
        RTS
*
MIXSE1  CMPB    #2              Mode 2 ?
        BNE     MIXSE2          Branch if no
        STB     FMODE           Save mode number
        LDA     SOLV
        ANDA    #3              Close product tank
        STA     SOLV            valves
        RTS
*
MIXSE2  CMPB    #3              Mode 3 ?
        BNE     MIXSE3          Branch if no
        STB     FMODE           Save mode number
        JSR     GETNUM          Get mixing ratio
        BNE     MIXS1           Branch if specified
        CLR     MRATIO
        LDA     SOLV
        ANDA    #$3             Close product tank
        STA     SOLV            valves
        BRA     MIXSE3          Disable mixing
MIXS1   STB     MRATIO          Save mixing ratio
        CLR     TIM3            Clear mixing timer
        JSR     VCLOSE          Set valves
        RTS
*
MIXSE3  CLR     FMODE           Disable feed mixing
        RTS
```

```
*
* Routine to initiate auto-update reports
*
AUTSET  LEAY    [-INSBUF-1,X]   Save report destination
        STY     AUACIA          ACIA address
        JSR     GETNUM          Get counter frequency
        BNE     AUT2            Branch if specified
AUT1    CLR     AUTOFG          Disable auto-reports
        RTS
AUT2    TSTB                    Is frequency negative ?
        BLE     AUT1            Branch if yes
        STB     REPTIM          Save frequency
        STB     AUTOFG          Enable auto-reports
        DECB                    Initialise auto-report
        STB     TIM1            timer
        RTS
*
* Routine to handle request for a single update report
*
REPSET  LEAX    [-INSBUF-1,X]   Save report destination
        STX     SRACIA          ACIA address
        LDA     #$FF
        STA     SRFLAG          Set report flag
        RTS
*
* Routine to change controller set point
*       Forms: C <controller> <new set point>
*              where <controller> options are :F,R,S
*
CONSET  LDA     0,X+            Get next instruction char
CONS1   CMPA    #'R             Is it "R" ?
        BEQ     CONS5           Branch if yes
        CMPA    #'S             Is it "S" ?
        BEQ     CONS2           Branch if yes
        CMPA    #'F             Is it "F" ?
        BEQ     CONS3           Branch if yes
CONS6   CMPA    #$0D            Is it <CR> ?
        BNE     CONSET          Loop if not
CONS7   RTS
*
CONS5   JSR     GETNUM          Get new reflux set point
        BEQ     CONS7           Do nothing if unspecified
        ANDA    #$0F            Limit to 0 -> $FFF
        STD     CNTLR2+SETPNT   Save it
        BRA     CONSET          Loop for more
CONS2   JSR     GETNUM          Get new steam set point
        BEQ     CONS7           Do nothing if unspecified
        STB     P5              Output it to steam loop
        BRA     CONSET          Loop for more
CONS3   JSR     GETNUM          Get new feed set point
        BEQ     CONS7           Do nothing if unspecified
        ANDA    #$0F            Limit to 0 -> $FFF
        STD     CNTLR1+SETPNT   Save it
        BRA     CONSET          Loop for more
*
* Routine to set/unset echo flag
*
ECHOST  JSR     YESNO           Scan for "Y" or "N"
        CMPA    #'N
        BEQ     ECHOS1          Branch if "N"
```

```
        CLR     ECHOFL          Enable echoing
        RTS
ECHOS1  STA     ECHOFL          Disable echoing
        RTS
*
* Routine to change integer format between decimal
* and hexadecimal
*
FORMST  JSR     YESNO           Scan for "Y" or "N"
        CMPA    #'N
        BEQ     FORMS1          Branch if "N"
        STA     FORMAT          Set flag for hexadecimal
        RTS
FORMS1  CLR     FORMAT          Set flag for decimal
        RTS
*
* Routine to enable/disable alarm checking
*
WARN    JSR     YESNO           Scan for "Y" or "N"
        CMPA    #'N
        BEQ     WARNS1          Branch if "N"
        CLR     ALARMF          Enable alarm checking
        CLR     ALARMC
        CLR     TIM5            Initialise alarm timer
        RTS
WARNS1  LDA     #$FF
        STA     ALARMF          Disable alarm checking
        RTS
*
* Routine to scan instruction buffer for "Y" or "N"
*
YESNO   LDA     ,X+             Get next char
        JSR     UPCASE          Change to upper case
        CMPA    #'Y             Is it "Y" ?
        BNE     YESN1           Branch if no
        RTS
YESN1   CMPA    #'N             Is it "N" ?
        BNE     YESN2           Branch if no
        RTS
YESN2   CMPA    #$0D            Is it <CR> ?
        BNE     YESNO           Loop if no
        TSTA                    Clear Z bit
        RTS
*
* Routine to shut down column
*
SHUTDN  LDA     #$FF            Disable alarm checking
        STA     ALARMF
        LDD     #50             Set feed and reflux pumps
        STD     CNTLR1+SETPNT   to slowly tick over to
        STD     CNTLR2+SETPNT   protect flowmeters
        LDX     #MESG1
        JSR     PSTRNG          Print first message
SHUT1   LDD     TEMP5           Wait for column to cool
        CMPD    #$10            by monitoring feed plate
        BGT     SHUT1           temperature
        LDX     #MESG2          Print second message
        JSR     PSTRNG
        RTS
*
```

```
MESG1   FCB     $0D,$0A,$0A
        FCC     'Turn off steam valve'
        FCB     $0D,$0A
        FCC     'Switch off tops and bottoms pumps'
        FCB     $0D,$0A
        FCC     'Turn off compressed air valve'
        FCB     $0D,$0A
        FCC     'Open air purge valve to reboiler chest'
        FCB     $0D,$0A
        FCC     'Wait for column to cool and next instruction'
        FCB     $0D,$8A
MESG2   FCB     $07,$0D,$0A
        FCC     'Switch off feed and reflux pumps'
        FCB     $0D,$0A
        FCC     'Turn off cooling water valves'
        FCB     $0D,$0A
        FCC     'Check that you are logged off VAX'
        FCB     $0D,$0A
        FCC     'And switch off column power supply'
        FCB     $0D,$8A
*
*********************************************************
* Get number from instruction buffer in current format
*
*    Entry: X - points to next char in buffer
*
*    Exit:  D - number in binary
*           X - point to last char used in buffer
*           C - Z bit set if no number i.e. end of data
*
*********************************************************
GETNUM  LDA     FORMAT          Decimal or hexadecimal ?
        BEQ     GETDEC          Branch if decimal
*********************************************************
* Get number from instruction buffer in hexadecimal
*
* Conditions: same as GETNUM
*
*********************************************************
GETHEX  PSHS    Y
        LDY     #0
GETHX1  JSR     INHEX           Remove leading non-hex chars
        BNE     GETHX2          Branch if valid hex number
        CMPA    #$0D            End of data ?
        BNE     GETHX1          Branch if not
        LEAX    -1,X            Point back to carriage return
        ORCC    #$04            Set Z bit
        PULS    Y,PC
GETHX2  EXG     D,Y             Shift binary number to D reg
        ASLB                    Shift four bits to right
        ROLA
        ASLB
        ROLA
        ASLB
        ROLA
        ASLB
        ROLA
        EXG     D,Y             Return binary number to Y reg
        LEAY    A,Y             Add on latest hex number
        JSR     INHEX           Get next char
```

```
        BNE     GETHX2          Branch if valid hex number
        EXG     D,Y             Put binary number into D reg
        LEAX    -1,X            Point back to final delimiter
        ANDCC   #$FB            Clear Z bit
        PULS    Y,PC
*******************************************************
* Get one character and convert to hexadecimal digit
*                                       if possible
*
*    Entry: X - points to character
*
*    Exit:  A - hex digit or non-hex ASCII char
*           X - incremented by one
*           C - Z bit set if non-hex ASCII char
*
*******************************************************
INHEX   LDA     ,X+             Get a char
        PSHS    A               Save it
        SUBA    #$30
        BMI     INHEX2          Go if char was < 30
        CMPA    #$09
        BLS     INHEX1          Go if char was between 30 & 39
        CMPA    #$10
        BLS     INHEX2          Go if char was between 3A & 40
        SUBA    #$07
INHEX1  CMPA    #$0F
        BLS     INHEX3          Go if char was between 41 & 46
INHEX2  LDA     ,S
INHEX3  CMPA    ,S+
        RTS
*******************************************************
* Get number from instruction buffer in decimal
*
* Conditions: same as GETNUM
*
*******************************************************
GETDEC  PSHS    Y
        LDY     #DECDIG         Clear storage area
GETDC4  CLR     ,Y+
        CMPY    #DECDIG+5
        BNE     GETDC4
GETDC1  JSR     INDEC           Remove leading non-dec chars
        BNE     GETDC2          Branch if valid dec digit
        CMPA    #$0D            End of data ?
        BNE     GETDC1          Branch if not
        LEAX    -1,X            Point back to carriage return
        ORCC    #$04            Set Z bit
        PULS    Y,PC            Return with Z bit set
GETDC2  LDY     #DECDIG         Shift digits one to right
GETDC3  LDB     1,Y
        STB     ,Y+
        CMPY    #DECDIG+4
        BNE     GETDC3
        STA     ,Y              Save new digit
        JSR     INDEC           Get next char
        BNE     GETDC2          Branch if valid dec digit
        JSR     CNVD2B
        LEAX    -1,X            Point back to final delimiter
        ANDCC   #$FB            Clear Z bit
        PULS    Y,PC
```

```
*********************************************************
* Get one character and convert to decimal digit
*                                 if possible
*
*    Entry: X - points to character
*
*    Exit:  A - decimal digit or non-dec ASCII char
*           X - incremented by one
*           C - Z bit set if non-dec ASCII char
*
*********************************************************
INDEC   LDA     ,X+             Get a char
        PSHS    A               Save it
        SUBA    #$30
        BMI     INDEC1          Go if char was < 30
        CMPA    #$09
        BLS     INDEC2          Go if char was between 30 & 39
INDEC1  LDA     ,S
INDEC2  CMPA    ,S+
        RTS
*********************************************************
* Convert 5 digit decimal number to 16 bit binary number
*
*    Entry: DECDIG - 5 decimal digits (1 digit/byte)
*
*    Exit:  D - binary number
*
*********************************************************
CNVD2B  PSHS    X,Y             Save registers
        LDX     #K10K           Point at constants
        LDY     #DECDIG         Point at decimal number
        CLRA                    Clear accumulator
        CLRB
CVDEC1  DEC     ,Y              Decrement decimal number
        BMI     CVDEC2          Branch if answer negative
        ADDB    1,X             Add constant to accumulator
        ADCA    ,X
        BRA     CVDEC1          Loop back
CVDEC2  LEAY    1,Y             Point at next digit
        LEAX    2,X             Point at next constant
        CMPX    #K10K+10        End of constant table ?
        BNE     CVDEC1          Branch if not
        PULS    X,Y,PC          Restore registers
*********************************************************
* Send number preceeded by a comma to the required serial
*                         device in the current format
*
*    Entry: D - number in binary
*           X - points to ACIA
*
*    Exit:  D - changed
*           X - points to ACIA
*
*********************************************************
SNDNUM  PSHS    D
        LDA     #',
        JSR     SEND            Send comma
        LDA     FORMAT          Decimal or hexadecimal ?
        PULS    D
        BEQ     SNDDEC          Branch if decimal
```

```
*********************************************************
* Send hexadecimal number to the required serial device
*
* Conditions: same as SNDNUM
*
*********************************************************
SNDHEX  CMPD    #$0             Is number zero ?
        LBEQ    SNZERO          Branch if yes
        PSHS    B,X             Save LS byte and ACIA address
        LDX     #SNDASC         Point at destination
        LDB     #$30
        STB     ,X+             Set first digit to zero
        JSR     SNDH1           Handle MS byte
        PULS    A               Retrieve LS byte
        JSR     SNDH1           Handle LS byte
        PULS    X               Restore ACIA address
        BRA     SNDDIG          Send 5 digit number
*
SNDH1   PSHS    A               Save byte
        LSRA
        LSRA                    Shift MS nibble to the
        LSRA                    right
        LSRA
        JSR     SNDH2           Handle MS nibble
        PULS    A               Retrieve byte
        ANDA    #$0F            Only keep LS nibble
        JSR     SNDH2           Handle LS nibble
        RTS
*
SNDH2   ADDA    #$30            Convert hex digit to
        CMPA    #$39            ASCII code
        BLS     SNDH3
        ADDA    #$07
SNDH3   STA     ,X+             Save digit at destination
        RTS
*********************************************************
* Send decimal number to the required serial device
*
* Conditions: same as SNDNUM
*
*********************************************************
SNDDEC  CMPD    #$0             Is number zero ?
        BEQ     SNZERO          Branch if yes
        JSR     CNVB2A          Convert bin to dec ASCII
        BRA     SNDDIG          Send 5 digit number
*********************************************************
* Convert 16 bit binary number to 5 digit ASCII encoded
*                                 decimal number
*
*    Entry: D - binary number
*
*    Exit : D - changed
*           SNDASC - 5 decimal digits (1 digit/byte)
*
*********************************************************
CNVB2A  PSHS    X,Y
        LDX     #K10K           Point to constants
        LDY     #SNDASC         Point to destination
CNVB1   CLR     DECHAR          Clear dec char
CNVB2   SUBB    1,X             Subtract two byte constant
```

```
        SBCA    ,X              From binary number
        BCS     CNVB3           Branch if overflow
        INC     DECHAR
        BRA     CNVB2
CNVB3   ADDB    1,X             Restore binary number to
        ADCA    ,X++            Prior last subtraction
        PSHS    A
        LDA     DECHAR          Get dec char
        ADDA    #$30            Convert to ASCII
        STA     ,Y+             And store
        PULS    A
        CMPX    #K10K+10        Finished ?
        BNE     CNVB1           Loop if not
        PULS    X,Y,PC
*********************************************************
* Send 5 digit ASCII encoded number to the required
* serial device but removing any leading zeros
*
*    Entry: SNDASC - 5 ASCII digits (1 digit/byte)
*
*********************************************************
SNDDIG  PSHS    Y               Save registers
        LDY     #SNDASC         Point at ASCII number
SNDD1   LDA     ,Y+             Get next character
        CMPA    #$30            Remove leading zeros
        BEQ     SNDD1
SNDD2   JSR     SEND            Send character
        LDA     ,Y+             Get next character
        CMPY    #SNDASC+6       All done ?
        BNE     SNDD2           Branch if not
        PULS    Y,PC            Restore registers and return
*********************************************************
* Send zero to required serial device
*********************************************************
SNZERO  LDA     #$30
        JSR     SEND
        RTS
*********************************************************
* Constants for decimal <-> hexadecimal conversions
*********************************************************
K10K    FDB     10000
        FDB     1000
        FDB     100
        FDB     10
        FDB     1
*
* Routine to change to upper case
*
UPCASE  CMPA    #$60            Is it lowercase letter
        BLE     UPCEND          Branch if not
        CMPA    #$7B
        BGE     UPCEND
        SUBA    #$20            Shift to uppercase
UPCEND  RTS
```

```
*********************************************************
* IRQ interrupt controller handler
*********************************************************
*
* Time to sample and control
*
IRQSER  CLR     SYSTAT
        JSR     READAD          Go to data aquisition
        INC     SYSTAT
        JSR     TIMER           Increment timers
        INC     SYSTAT
        JSR     CHECK           Check alarms
        INC     SYSTAT
        JSR     CONTRL          Go to control routine
        INC     SYSTAT
        JSR     PUMPS           Implement control
        INC     SYSTAT
        JSR     FILTER          Filter measurements
        INC     SYSTAT
        JSR     UPDATE          Send update report
        INC     SYSTAT
        JSR     FNMIX           Control feed mixing
        CLR     SYSTAT
        RTI
*
* Data aquisition routine
*
READAD  LDD     #997            Reset for 1sec interrupt
        STD     COUNT5
        LDX     #FLOW           Point at flow sensor data
        LDY     #COUNT1         Read flow sensor 1
        JSR     DAQ6
        LDY     #COUNT2         Read flow sensor 2
        JSR     DAQ6
        LDY     #COUNT3         Read flow sensor 3
        JSR     DAQ6
        LDY     #COUNT4         Read flow sensor 4
        JSR     DAQ6
*
        LDA     CSTAT           Increment pointer offset
        INCA                    for capacitance cell data
        ANDA    #$7
        STA     CSTAT
*
        LDX     #CELL1          Point at capacitance cell data
        LDY     #COUNT6         Read capacitance cell 1
        JSR     DAQ8
        LDX     #CELL2
        LDY     #COUNT7         Read capacitance cell 2
        JSR     DAQ8
        LDX     #CELL3
        LDY     #COUNT8         Read capacitance cell 3
        JSR     DAQ8
        LDX     #CELLR
        LDY     #COUNT9         Read capacitance cell 4
        JSR     DAQ8
        BRA     DAQ7            Skip over subroutines
```

```
*
* Routine to read and process capacitance cell data
*
DAQ8    LDD     ,Y              Get counter reading
        COMA                    Convert from residual to
        ANDA    #$7F            actual count
        COMB
        PSHS    D               Save count
        LDD     #$FFFF
        STD     ,Y              Reset counter
        PULS    Y               Retrieve count
        LDA     CSTAT           Get offset pointer
        LSLA                    Manipulate offset
        LSLA
        ADDA    #2
        STY     A,X             Store count at offset
        RTS
*
* Routine to read and process flow sensor data
*
DAQ6    LDD     ,Y              Get counter reading
        COMA                    Convert from residual to
        COMB                    actual count
        STD     ,X++            Store count
        LDD     #$FFFF
        STD     ,Y              Reset counter
        RTS
*
DAQ7    LDX     #CELL1          Average cell 1 readings
        JSR     DAQ9
        LDX     #CELL2          Average cell 2 readings
        JSR     DAQ9
        LDX     #CELL3          Average cell 3 readings
        JSR     DAQ9
        LDX     #CELLR          Average cell 4 readings
        JSR     DAQ9
        BRA     DAQ10           Skip over subroutines
*
NUMAVG  FDB     0,4
*
* Routine to average last 8 cell readings
*
DAQ9    LDB     #7              Set counter for 8 readings
        JSR     PUSH32          Push first reading on stack
DAQ12   LEAX    4,X             Point at next reading
        JSR     PUSH32          Push it on stack
        JSR     DADD            Add it to total
        DECB                    All 8 readings done ?
        BNE     DAQ12           Branch if not
        PSHS    X               Save address
        LDX     #NUMAVG         Point at constant
        JSR     PUSH32          Push 4 on stack
        JSR     DDIV            Divide total by 4
        PULS    X               Retrieve address
        LEAX    4,X             Get result address
        JSR     PULL16
        JSR     PULL16          Save result
        RTS
```

```
*
* Routine read the data acquision unit inputs
*
DAQ10   CLR     DAMERR          Clear error counter
        CLRB                    First channel pointer
        LDX     #DAM            Set up data area pointer
        LDA     DLSB            Read periph reg to clear any EOC
        LDA     #$36            Pulse the start line
        STA     DCSR2
        NOP
        NOP
        LDA     #$3E
        STA     DCSR2
DAQ1    LDA     DCSR2           Wait for EOC
        BMI     DAQ1A
        LDY     COUNT5          Check if >1/2sec
        CMPY    #500
        BGT     DAQ1            Loop if OK
        LDA     #$FF            Error - missed EOC'S
        STA     DAMERR          Indicate error
        BRA     DAQ20
DAQ1A   LDA     DMSB            Read MSB
        PSHS    A               Save MSB for later
        LSRA
        LSRA
        LSRA
        LSRA
        PSHS    B               Is it correct ?
        CMPA    ,S+
        BEQ     DAQ2            Yes
        INC     DAMERR          No
        PULS    A               Restore stack
        BRA     DAQ3
DAQ2    PULS    A               Retrieve MSB from stack
        ANDA    #$0F            Strip channel no
        STA     ,X              Store it
        LDA     DLSB            Get LSB
        STA     1,X             Store it
DAQ3    INCB                    Increment channel pointer
        LEAX    2,X             Increment data pointer
        CMPB    #$10            Last channel ?
        BLO     DAQ1            No
DAQ20   RTS
*
* Increment software timers
*
TIMER   LDX     #TIM1           Point to counters
TIMER1  INC     ,X+             Increment counter
        CMPX    #TIM5+1         Last counter ?
        BNE     TIMER1          No
        RTS
*
* Check for any alarm conditions and warn the operator every
* 10 seconds if necessary.
*
CHECK   LDA     #$34
        STA     P4CSR           Switch beeper off
        LDA     ALARMF          Checking enabled ?
        BNE     CHEND2          Return if no
        LDA     TIM5            Time for checking
```

```
        CMPA    #10             (i.e. every 10 sec)
        BNE     CHEND           Return if no
        CLR     ALARMC          Clear alarm condition flag
        CLR     TIM5
*
        LDD     REBLVL          Check reboiler level
        CMPD    #$190           Too low ?
        BGT     CHEC1           Branch if OK
        LDX     #AMESG1         Print warning message
        JSR     CHEMES
        BRA     CHEC2
CHEC1   CMPD    #$310           Too high ?
        BLT     CHEC2           Branch if OK
        LDX     #AMESG2         Print warning message
        JSR     CHEMES
*
CHEC2   LDD     REFLVL          Check reflux level
        CMPD    #$740           Too low ?
        BGT     CHEC3           Branch if OK
        LDX     #AMESG3         Print warning message
        JSR     CHEMES
        BRA     CHEC4
CHEC3   CMPD    #$8C0           Too high ?
        BLT     CHEC4           Branch if OK
        LDX     #AMESG4         Print warning message
        JSR     CHEMES
*
CHEC4   LDD     TCELL2          Check cell temperature
        CMPD    #1230           Greater than 30 deg C
        BLT     CHEND           Branch if no
        LDX     #AMESG5         Print warning message
        JSR     CHEMES
*
CHEND   LDA     TIM5
        BITA    #$1             Time to switch on beeper ?
        BEQ     CHEND2          Branch if no
        LDA     ALARMC          Alarm condition existing ?
        BEQ     CHEND2          Branch if no
        LDA     #$3C            Switch on beeper
        STA     P4CSR
CHEND2  RTS
*
CHEMES  JSR     PSTRNG          Print warning to screen
        LDA     #$FF            Set alarm condition flag
        STA     ALARMC
        RTS
*
AMESG1  FCB     $0D,$0A
        FCC     'WARNING - low reboiler level'
        FCB     $0D,$8A
AMESG2  FCB     $0D,$0A
        FCC     'WARNING - high reboiler level'
        FCB     $0D,$8A
AMESG3  FCB     $0D,$0A
        FCC     'WARNING - low reflux accumulater level'
        FCB     $0D,$8A
AMESG4  FCB     $0D,$0A
        FCC     'WARNING - high reflux accumulater level'
        FCB     $0D,$8A
AMESG5  FCB     $0D,$0A
```

```
        FCC     'WARNING - high bottoms product temperature'
        FCB     $0D,$8A
*
* Control routine algorithm is
*
*    dV = K1*(SP-PV(t))+K2*(SP-PV(t-1))
*    VALVE = LAST VALVE+dV
*    VNEW = VALVE/16 (8 BITS)
*
CONTRL  LDY     #CNTLR1         Calculate controller 1
        JSR     CONT1
        LDY     #CNTLR2         Calculate controller 2
        JSR     CONT1
        LDY     #CNTLR3         Calculate controller 3
        JSR     CONT1
        LDY     #CNTLR4         Calculate controller 4
        JSR     CONT1
        RTS
*
CONT1   LDD     ERPRES,Y        Shift present error to
        STD     ERPREV,Y        previous error variable
        LDD     SETPNT,Y        Get set point
        SUBD    [PROVAR,Y]      Subtract process variable
        STD     ERPRES,Y        Save result as present error
        LEAX    ERPREV,Y
        JSR     PUSH16          Push previous error on stack
        JSR     FLTS
        LEAX    COEF2,Y
        JSR     PUSH32          Push K2 on stack
        JSR     FMUL            Multiply them
        LEAX    ERPRES,Y
        JSR     PUSH16          Push present error on stack
        JSR     FLTS
        LEAX    COEF1,Y
        JSR     PUSH32          Push K1 on stack
        JSR     FMUL            Multiply them
        JSR     FADD            Add products to give dV
        JSR     FIXD
        LEAX    CONOUT,Y
        JSR     PUSH32          Push VALVE on stack
        JSR     DADD            Add dV to give VNEW
        JSR     PULL32          Pull new VALVE from stack
        JSR     LIMIT           Limit to range 0 to $FFF
        RTS
*
* Routine to limit controller outputs to range 0 to $FFF
*
LIMIT   LDD     CONOUT,Y        Get MS 16 bits
        BMI     LIM1            Branch if < 0
        BNE     LIM2            Branch if > $FFFF
        LDD     CONOUT+2,Y      Get LS 16 bits
        CMPD    #$FFF
        BGT     LIM2            Branch if > $FFF
        CMPD    #$0
        BLT     LIM2            Branch if < 0
        RTS
LIM2    LDD     #$0             Set controller output
        STD     CONOUT,Y        to $FFF
        LDD     #$FFF
        STD     CONOUT+2,Y
```

```
        RTS
LIM1    LDD     #$0             Zero controller output
        STD     CONOUT,Y
        STD     CONOUT+2,Y
        RTS
*
* Routine to send controller outputs to pumps
*
PUMPS   LDY     #CNTLR1         Action controller 1
        JSR     PUMPS1
        LDY     #CNTLR2         Action controller 2
        JSR     PUMPS1
        LDY     #CNTLR3         Action controller 3
        JSR     PUMPS1
        LDY     #CNTLR4         Action controller 4
        JSR     PUMPS1
        RTS
*
PUMPS1  LDD     CONOUT+2,Y      Get controller output
        LSRA                    Change from 0-$FFF range
        RORB                    to 0-$FF range
        LSRA
        RORB
        LSRA
        RORB
        LSRA
        RORB
        PSHS    B
        LDA     [CONVAR,Y]      Get present pump position
        STA     PMPTMP+1        Save it
        CLRA
        STA     PMPTMP          Clear MS byte
        SUBD    PMPTMP          Up or down ?
        BGE     PUMPS3          Jump if up or same
        ADDD    SLEWDN,Y
        PULS    B
        BGE     PUMPS2          Jump if < slew limit
        LDD     PMPTMP
        SUBD    SLEWDN,Y        Set at limit
        BRA     PUMPS2
PUMPS3  SUBD    SLEWUP,Y
        PULS    B
        BLE     PUMPS2          Jump if < slew limit
        LDD     PMPTMP
        ADDD    SLEWUP,Y        Set at limit
PUMPS2  STB     [CONVAR,Y]      Set new pump position
        RTS
*
* First order digital filtering routine
*
*       Y(I+1)=Y(I)+(U(I+1)-Y(I+1))/TIMCON
*
FILTER  LDY     #FILDAT         Point to filter data
        LDB     #4              Get filter count
FILTR1  JSR     FILTR2          Action filter
        LEAY    10,Y            Point to next filter
        DECB                    All done ?
        BNE     FILTR1          Branch if no
        RTS
*
```

```
FILTR2  LDX     MEASUR,Y
        JSR     PUSH16          Push measurement on stack
        JSR     FLTS
        LDX     FILT32,Y        Push old filter output
        JSR     PUSH32          on stack
        JSR     FSUB            Get the difference
        JSR     UNDFLO          Test for underflow
        LEAX    TIMCON,Y
        JSR     PUSH32          Push time constant on stack
        JSR     FDIV            Divide them
        JSR     UNDFLO          Test for underflow
        LDX     FILT32,Y        Push old filter output
        JSR     PUSH32          on stack
        JSR     FADD            Add to them
        JSR     UNDFLO          Test for underflow
        JSR     PTOF
        JSR     PULL32
        JSR     FIXS            Change to single precision
        LDX     FILOUT,Y
        JSR     PULL16          Save new filter output
        RTS
*
UNDFLO  ANDA    #$1E            Check for under flow
        BEQ     UNDFL1          Branch if OK
        PSHS    X
        LDX     #FLZERO         Push zero onto APU stack
        JSR     PUSH32
        PULS    X
UNDFL1  RTS
*
FLZERO  FDB     {0.0}
*
* Update report routine
*
UPDATE  LDA     AUTOFG          Auto-update active ?
        BEQ     UPD1            Branch if no
        LDA     TIM1
        CMPA    REPTIM          Time to report ?
        BLT     UPD1            Branch if no
        CLR     TIM1
        LDX     AUACIA          Point at destination ACIA
        JSR     REPORT          Send auto-update report
UPD1    LDA     SRFLAG          Single report requested ?
        BEQ     UPD2            Branch if no ?
        CLR     SRFLAG
        LDX     SRACIA          Point at destination ACIA
        JSR     REPORT          Send single report
UPD2    RTS
*
REPORT  LDA     #INSCHR         Send hash <#>
        JSR     SEND
        LDA     #'I             Send "I" for echo ignore
        JSR     SEND
        LDY     #PRODAT         Point at filtered data
        LDA     #4              Number of channels
REPOR1  PSHS    A
        LDD     ,Y++            Get filtered number
        JSR     SNDNUM          Send if
        PULS    A
        DECA                    All filtered data sent ?
```

```
         BNE     REPOR1          Branch if no
         LDD     TCELL1          Report cell 1 temperature
         JSR     SNDNUM
         LDD     TCELL2          Report cell 2 temperature
         JSR     SNDNUM
         LDD     TCELL3          Report cell 3 temperature
         JSR     SNDNUM
         LDD     CELL1+32        Report averaged cell 1 freq
         JSR     SNDNUM
         LDD     CELL2+32        Report averaged cell 2 freq
         JSR     SNDNUM
         LDD     CELL3+32        Report averaged cell 3 freq
         JSR     SNDNUM
REPOR6   CMPX    #TERMNL
         BNE     REPOR5          If reporting to terminal
         JSR     CRLF            send <CR><LF> pair
         RTS
REPOR5   LDA     #$0D
         JSR     SEND            Else only <CR>
         RTS
*
* Feed mixing routine
*
FNMIX    LDA     FMODE           Get feed mixing mode
         CMPA    #1
         BEQ     FMIX5           Branch if mode 1
         CMPA    #2
         BEQ     FMIX            Branch if mode 1
         CMPA    #3
         BEQ     FMIX3           Branch if mode 1
         RTS                     Else do nothing
*
* Feed mixing mode 1
*
FMIX5    LDA     TIM4            Only do routine 2 seconds
         ANDA    #$E             in every 16
         BNE     FCLOSE          Branch if not time
         LDD     TCELL1          Get feed temperature
         SUBD    #614            Subtract 15 deg C
         STD     COMPBF
         LDX     #COMFAC         Multiply the difference
         JSR     PUSH32          by the compensation
         LDX     #COMPBF         factor
         JSR     PUSH16
         JSR     FLTS
         JSR     FMUL
         JSR     FIXS
         JSR     PULL16          Pull the result from stack
         LDD     COMPBF          and add it to feed composition
         ADDD    CELL1+32
         CMPD    #28998          Above 50 mole percent ?
         BLO     FOPEN           Branch if not
         LDA     #$33            Open valves from bottoms tank
         STA     SOLV
         RTS
*
FOPEN    LDA     #$F             Open valves from tops tank
         STA     SOLV
         RTS
*
```

```
FCLOSE  LDA     #$3             Close all valves from
        STA     SOLV            product tanks
        RTS
*
COMFAC  FDB     {-1.7827}
*
* Feed mixing mode 2
*
FMIX    LDA     TIM4            Is 256 seconds up ?
        BNE     FMIX4           Branch if not
        LDA     FMTIM           Increment slow counter
        INCA
        STA     FMTIM
        CMPA    #5              Slow counter at 5*256 seconds ?
        BLT     FMIX4           Branch if not
        CLR     FMTIM           Reset slow timer
        LDA     SOLV            Open valves from product tanks to
        ORA     #$3C            feed tanks
        STA     SOLV
FMIX4   LDD     DAM+24          Product tank empty ?
        CMPD    #$140           Branch if not
        BGT     FMIX3
        LDD     DAM+26          Product tank empty ?
        CMPD    #$140           Branch if not
        BGT     FMIX3
        LDA     SOLV            Close valves from product tanks to
        ANDA    #$3             feed tanks
        STA     SOLV
        RTS
*
* Feed mixing mode 3
*
FMIX3   LDA     TIM3            Get mixing timer
        CMPA    #1              Is it 1 ?
        BNE     FMIX1           Branch if no
        LDA     SOLV
        ORA     #$3C            Open all valves from
        STA     SOLV            product tanks
        RTS
FMIX1   CMPA    MRATIO          If timer equals ratio
        BNE     FMIX2           then clear timer
        CLR     TIM3
VCLOSE  LDA     SOLV
        ANDA    #$27            Open one valve from
        STA     SOLV            each product tank
FMIX2   RTS
*
* Filter data
*
FILDAT  FDB     PRODAT          Filter number 1
        FDB     TEMP1           Measured variable address
        FDB     F32DAT          Output variable address
        FDB     {10.0}
        FDB     PRODAT+2        Filter number 2
        FDB     TEMP8
        FDB     F32DAT+4
        FDB     {10.0}
        FDB     PRODAT+4        Filter number 3
        FDB     TOPFLO
        FDB     F32DAT+8
```

```
        FDB     {10.0}
        FDB     PRODAT+6        Filter number 4
        FDB     BOTFLO
        FDB     F32DAT+12
        FDB     {10.0}
*
* RAM initialisation data
*
RAMINT  FCB     RAM1-RAMINT-3   Terminal ACIA buffer data
        FDB     TERBUF
        FDB     TERMNL
        FDB     COMPUT
RAM1    FCB     RAM2-RAM1-3
        FDB     TERBUF+RECPTR
        FDB     $0
RAM2    FCB     RAM3-RAM2-3
        FDB     TERBUF+INSPTR
        FDB     TERBUF+INSBUF
RAM3    FCB     RAM4-RAM3-3     Computer ACIA buffer data
        FDB     COMBUF
        FDB     COMPUT
        FDB     TERMNL
RAM4    FCB     RAM5-RAM4-3
        FDB     COMBUF+RECPTR
        FDB     $0
RAM5    FCB     RAM6-RAM5-3
        FDB     COMBUF+INSPTR
        FDB     COMBUF+INSBUF
RAM6    FCB     RAM7-RAM6-3     Feed flow controller data
        FDB     CNTLR1
        FDB     FEDFLO
        FDB     P1
        FDB     6,6
        FDB     428
        FDB     {3.6}
        FDB     {-3}
RAM7    FCB     RAM8-RAM7-3     Reflux flow controller data
        FDB     CNTLR2
        FDB     REFFLO
        FDB     P2
        FDB     10,10
        FDB     279
        FDB     {3.6}
        FDB     {-3}
RAM8    FCB     RAM9-RAM8-3     Reflux accumulator level
        FDB     CNTLR3          controller data
        FDB     REFLVL
        FDB     P3
        FDB     10,10
        FDB     $800
        FDB     {-10.05555}
        FDB     {10}
RAM9    FCB     RAM10-RAM9-3    Reboiler level controller data
        FDB     CNTLR4
        FDB     REBLVL
        FDB     P4
        FDB     10,10
        FDB     $250
        FDB     {-10.02777}
        FDB     {10}
```

```
RAM10   FCB     RAM11-RAM10-3   Solenoid manifold sequence
        FDB     RESEQ
*       FCB     0,1,2,3,0,1,2,3,0,1,2,3
        FCB     3,3,3,3,3,3,3,3,3,3,3,3
        FCB     1,120
RAM11   FCB     $0
        END
```

A.5.3 APU DRIVER

```
        NAM     AM9511
        OPT     NOG,LLE=120,P=62
*
* Driver for the AM9511 arithmetic processor unit
*
* Code is resident in an onboard 2708 EPROM
*
STACK   EQU     $CFFF   APU stack address
APUCSR  EQU     $CFFE   Control/status register
*
        ORG     $6000
SADD    LDA     #$6C
        JMP     COMAND
SSUB    LDA     #$6D
        JMP     COMAND
SMUL    LDA     #$6E
        JMP     COMAND
SMUU    LDA     #$76
        JMP     COMAND
SDIV    LDA     #$6F
        JMP     COMAND
DADD    LDA     #$2C
        JMP     COMAND
DSUB    LDA     #$2D
        JMP     COMAND
DMUL    LDA     #$2E
        JMP     COMAND
DMUU    LDA     #$36
        JMP     COMAND
DDIV    LDA     #$2F
        JMP     COMAND
FADD    LDA     #$10
        JMP     COMAND
FSUB    LDA     #$11
        JMP     COMAND
FMUL    LDA     #$12
        JMP     COMAND
FDIV    LDA     #$13
        JMP     COMAND
SQRT    LDA     #$1
        JMP     COMAND
SIN     LDA     #$2
        JMP     COMAND
COS     LDA     #$3
        JMP     COMAND
TAN     LDA     #$4
        JMP     COMAND
ASIN    LDA     #$5
        JMP     COMAND
ACOS    LDA     #$6
        JMP     COMAND
```

```
ATAN    LDA     #$7
        JMP     COMAND
LOG     LDA     #$8
        JMP     COMAND
LN      LDA     #$9
        JMP     COMAND
EXP     LDA     #$A
        JMP     COMAND
PWR     LDA     #$B
        JMP     COMAND
NOP     LDA     #$D
        JMP     COMAND
FIXS    LDA     #$1F
        JMP     COMAND
FIXD    LDA     #$1E
        JMP     COMAND
FLTS    LDA     #$1D
        JMP     COMAND
FLTD    LDA     #$1C
        JMP     COMAND
CHSS    LDA     #$74
        JMP     COMAND
CHSD    LDA     #$34
        JMP     COMAND
CHSF    LDA     #$15
        JMP     COMAND
PTOS    LDA     #$77
        JMP     COMAND
PTOD    LDA     #$37
        JMP     COMAND
PTOF    LDA     #$17
        JMP     COMAND
POPS    LDA     #$78
        JMP     COMAND
POPD    LDA     #$38
        JMP     COMAND
POPF    LDA     #$18
        JMP     COMAND
XCHS    LDA     #$79
        JMP     COMAND
XCHD    LDA     #$39
        JMP     COMAND
XCHF    LDA     #$19
        JMP     COMAND
PUPI    LDA     #$1A
*
* COMMAND APU ,WAIT UNTIL DONE & RETURN STATUS IN "A"
*
COMAND  STA     APUCSR
COMAN1  LDA     APUCSR
        BMI     COMAN1
        RTS
```

```
*
* PUSH NUMBER POINTED TO BY "X" ON TO STACK
*
PUSH32  PSHS    A
        LDA     3,X
        STA     STACK
        LDA     2,X
        STA     STACK
        PULS    A
PUSH16  PSHS    A
        LDA     1,X
        STA     STACK
        LDA     ,X
        STA     STACK
        PULS    A,PC
*
* PULL NUMBER OFF STACK TO WHERE POINTED BY "X"
*
PULL32  PSHS    A
        LDA     STACK
        STA     ,X
        LDA     STACK
        STA     1,X
        LDA     STACK
        STA     2,X
        LDA     STACK
        STA     3,X
        PULS    A,PC
*
PULL16  PSHS    A
        LDA     STACK
        STA     ,X
        LDA     STACK
        STA     1,X
        PULS    A,PC
        END
```

A.5.4 TANK VOLUME TABLE

```
         NAM      TANK
         OPT      NOG,LLE=120,P=62
         ORG      $2800
*
* LOOKUP TABLE RELATING LIQUID DEPTH IN FEED TANKS
* TO LIQUID VOLUME
*
* 0 TO 20
*
         FCB      $0,$0,$0,$1,$1,$1
         FCB      $2,$2,$3,$3,$4
         FCB      $4,$5,$6,$6,$7
         FCB      $8,$9,$9,$10,$11
*
* 21 TO 40
*
         FCB      $12,$12,$13,$14,$15
         FCB      $16,$17,$18,$19,$20
         FCB      $20,$21,$22,$23,$24
         FCB      $25,$26,$27,$28,$29
*
* 41 TO 60
*
         FCB      $30,$31,$32,$33,$34
         FCB      $35,$36,$37,$38,$39
         FCB      $40,$42,$43,$44,$45
         FCB      $46,$47,$48,$49,$50
*
* 61 TO 80
*
         FCB      $51,$52,$53,$54,$55
         FCB      $56,$57,$58,$60,$61
         FCB      $62,$63,$64,$65,$66
         FCB      $67,$68,$69,$70,$71
*
* 81 TO 100
*
         FCB      $72,$73,$74,$75,$76
         FCB      $77,$78,$79,$80,$80
         FCB      $81,$82,$83,$84,$85
         FCB      $86,$87,$88,$88,$89
*
* 101 TO 120
*
         FCB      $90,$91,$91,$92,$93
         FCB      $94,$94,$95,$96,$96
         FCB      $97,$97,$98,$98,$99
         FCB      $99,$99,$FF,$FF,$FF
*
         END
```

APPENDIX B

OPERATOR INTERFACE

B.1 COLUMN PROGRAM LISTING

```
      PROGRAM COLUMN
C
C A program to interactively run the 9" distillation column. It
C assumes the column microprocessor is running the COLSYS program.
C It allows the operator to monitor the column's performance, to
C change the column inputs, or to read the column inputs from a
C data file and write the results to a data file.
C
C              P.W.M.Janssen 1986
C
      COMMON/ COL1 / TEMP1,TEMP8,TOPCOMP,BOTCOMP,FEEDCOMP
      COMMON/ COL2 / VOLTOP,VOLBOT,VOLREF,VOLFEED,STMFLOW
      COMMON/ COL3 / FMASSTOP,FMASSBOT,FMASSREF,FMASSFEED
      COMMON/ COL4 / FMOLETOP,FMOLEBOT,FMOLEREF,FMOLEFEED
      COMMON/ COL5 / TIME,ESCFLG,ICHAN
      LOGICAL ESCFLG
      CHARACTER FILNAM*40,HEADING*80
C
C Switch alarms off during initial interactive session
C
      TYPE *,'#W N'
C
C Get initial column inputs and send them to the micro
C
      WRITE (6,110)
110   FORMAT(/' Enter feed flow rate in l/min ? [1.75] ',$)
      READ(5,140) VOLFEED
140   FORMAT(F10.0)
      IF (VOLFEED .EQ. 0.0) VOLFEED = 1.75
C
      WRITE (6,120)
120   FORMAT(/' Enter reflux flow rate in l/min ? [0.95] ',$)
      READ(5,140) VOLREF
      IF (VOLREF .EQ. 0.0) VOLREF = 0.95
C
      WRITE (6,130)
130   FORMAT(/' Enter steam condensate flow rate in l/min ',
     1        '? [1.1] ',$)
      READ(5,140) STMFLOW
      IF (STMFLOW .EQ. 0.0) STMFLOW = 1.1
C
      CALL Control_out
C
C Switch alarms on and screen echo off
C
      TYPE *,'#W Y'
      TYPE *,'#E N'
```

```
C
C Prompt operator with main menu
C
60      INS = 0
        WRITE(6,300)
300     FORMAT(/,20X,'1) Show column''s current state',/,
        1         20X,'2) Continuous display of column state',/,
        2         20X,'3) Change column inputs',/,
        3         20X,'4) Initiate input from file',/,
        4         20X,'5) Exit to operating system',//,
        5         ' Enter option ? [1] ',$)
        READ(5,165)INS
165     FORMAT(I10)
        GO TO (10,20,30,40,50), INS
C
C Show column's current state
C
10      CALL Data_aquisition
        CALL Screen_write
        GO TO 60
C
C Continuously update column's current state
C
20      CALL Data_aquisition
        CALL Screen_write
        WRITE(6,123) CHAR(27)
        CALL Esc_set
70      CALL Wait(10.0)
        CALL Data_aquisition
        CALL Screen_write
        WRITE(6,123) CHAR(27)
        IF (.NOT.ESCFLG) GO TO 70
        CALL Esc_clear
        GO TO 60
123     FORMAT('+',A1,'Y6 Hit <ESC> to return to menu',$)
C
C Change column inputs
C
30      WRITE(6,900)
900     FORMAT(//,20X,'1) Feed flow rate',/,
        2          20X,'2) Reflux flow rate',/,
        3          20X,'3) Steam condensate flow rate',//,
        4      ' Which variable do you which to change ? [none] ',$)
        READ(5,165)INS
        GO TO (41,42,43),INS
        GO TO 60
41      WRITE(6,810) VOLFEED
810     FORMAT(' Old value - ',F6.3,/,' Enter new value - ',$)
        READ(5,*) VOLFEED
        GO TO 44
42      WRITE(6,810) VOLREF
        READ(5,*) VOLREF
        GO TO 44
43      WRITE(6,810) STMFLOW
        READ(5,*)STMFLOW
44      CALL Control_out
        GO TO 60
```

```
C
C Read inputs from data file and write outputs to new file
C
40      WRITE(6,210)
210     FORMAT(/' Enter input data file name - ',$)
        READ(5,200) FILNAM
200     FORMAT(A)
        OPEN(UNIT=11,FILE=FILNAM,STATUS='OLD',ERR=40)
        WRITE(6,220)
220     FORMAT(/' Enter output data file name - ',$)
        READ(5,200) FILNAM
        OPEN(UNIT=10,FILE=FILNAM,STATUS='NEW')
C
C Write information header to output file
C
        WRITE (6,*)' Enter file discription (finish with EXIT)'
180     READ(5,201) NCHRS,HEADING
201     FORMAT(Q,A)
        IF(HEADING(1:4).NE.'EXIT'.AND.HEADING(1:4).NE.'exit') THEN
           WRITE(10,600) HEADING(1:NCHRS)
600        FORMAT('C ',A)
           GO TO 180
        END IF
C
C Initialise system
C
        TIME = 0
        CALL Data_aquisition
        CALL File_write
        READ(11,*,END=191)VOLFEED,VOLREF,STMFLOW,FDUMMY
        CALL Control_out
        CALL Data_aquisition
        CALL File_write
        CALL Screen_write
        CALL Esc_set
C
C Start of Wakevery/Sleep loop
C
        CALL Wakevery(240.0)
190     CALL Sleep
        CALL Data_aquisition
        READ(11,*,END=191)VOLFEED,VOLREF,STMFLOW,FDUMMY
        CALL Control_out
        TIME = TIME + 4.0
        CALL File_write
        CALL Screen_write
        IF (.NOT.ESCFLG) GO TO 190
C
C End of loop, clean up and return to main MENU
C
191     CALL Esc_clear
        CALL Wakevery(0.0)
        CLOSE (10)
        CLOSE (11)
        GO TO 60
C
50      STOP 'Exiting COLUMN'
C
        END
```

```
          SUBROUTINE File_write
C
C Routine to write column operating data to file
C
         COMMON/ COL1 / TEMP1,TEMP8,TOPCOMP,BOTCOMP,FEEDCOMP
         COMMON/ COL2 / VOLTOP,VOLBOT,VOLREF,VOLFEED,STMFLOW
         COMMON/ COL3 / FMASSTOP,FMASSBOT,FMASSREF,FMASSFEED
         COMMON/ COL4 / FMOLETOP,FMOLEBOT,FMOLEREF,FMOLEFEED
         COMMON/ COL5 / TIME,ESCFLG,ICHAN
         LOGICAL ESCFLG
C
         WRITE(10,500,ERR=10)TIME,VOLFEED,VOLREF,STMFLOW,
         1        TOPCOMP,BOTCOMP,FEEDCOMP,
         2        FMOLETOP,FMOLEBOT,
         3        TEMP1,TEMP8
500      FORMAT(' ',F7.1,2X,F5.3,2X,F5.3,2X,F5.3,
         1        2X,F8.6,2X,F8.6,2X,F6.4,
         2        2X,F5.2,2X,F5.2,
         3        2X,F5.2,2X,F5.2)
C
         RETURN
C
10       TYPE *,'File_write conversion error'
C
         RETURN
         END
```

```
          SUBROUTINE Screen_write
C
C Routine to write column operating data to terminal
C
        COMMON/ COL1 / TEMP1,TEMP8,TOPCOMP,BOTCOMP,FEEDCOMP
        COMMON/ COL2 / VOLTOP,VOLBOT,VOLREF,VOLFEED,STMFLOW
        COMMON/ COL3 / FMASSTOP,FMASSBOT,FMASSREF,FMASSFEED
        COMMON/ COL4 / FMOLETOP,FMOLEBOT,FMOLEREF,FMOLEFEED
        COMMON/ COL5 / TIME,ESCFLG,ICHAN
        LOGICAL ESCFLG
C
C Erase screen and move cursor to top left hand corner
C
        CALL LIB$ERASE_PAGE(1,1)
C
        REFRATIO=FMOLEREF/FMOLETOP
C
        WRITE(6,400,ERR=10)TIME,TEMP1,TEMP8,
        1       VOLFEED,VOLTOP,VOLBOT,VOLREF,
        2       FMOLEFEED,FMOLETOP,FMOLEBOT,FMOLEREF,
        3       FEEDCOMP,TOPCOMP,BOTCOMP,
        4       STMFLOW,REFRATIO
400     FORMAT('+Time ',F8.2,' minutes'//
        1    ' Temperatures:  tray 1  ',F5.2,5X,'tray 8  ',F5.2,//,
        2    ' Volume flows:  feed',F7.3,5X,'top',F7.3,5X,'bot',
        3       F7.3,5X,'ref',F7.3,//,
        4    ' Molar flows:   feed',F7.2,5X,'top',F7.2,5X,'bot',
        5       F7.2,5X,'ref',F7.2,//,
        6    ' Compositions:  feed',F7.3,5X,'top',F7.3,5X,'bot',
        7       F7.3,//,
        8    ' Steam flow  ',F7.3,//,
        9    ' Reflux ratio  ',F6.3)
        RETURN
C
10      TYPE *,'Screen_write conversion error'
        RETURN
C
        END

          SUBROUTINE Data_aquisition
C
C Routine prompts the microprocessor for a single reading, and then
C processes the response to meaningfull physical units using the
C calibrations provided.
C
        COMMON/ COL1 / TEMP1,TEMP8,TOPCOMP,BOTCOMP,FEEDCOMP
        COMMON/ COL2 / VOLTOP,VOLBOT,VOLREF,VOLFEED,STMFLOW
        COMMON/ COL3 / FMASSTOP,FMASSBOT,FMASSREF,FMASSFEED
        COMMON/ COL4 / FMOLETOP,FMOLEBOT,FMOLEREF,FMOLEFEED
        CHARACTER DIS*3
C
C Prompt microprocessor for a report of column outputs
C
        WRITE(6,300)
300     FORMAT(' #R')
C
C Accept report
C
        READ(5,123)DIS,IT1,IT8,IFT,IFB,ITC1,ITC2,
        1                                  ITC3,IFC1,IFC2,IFC3
```

```
123     FORMAT(A3,10I6)
C
C Scale raw data
C
        TEMP1 = 60.0+40.0*(IT1/4095.0)
        TEMP8 = 60.0+40.0*(IT8/4095.0)
        TCELL1 = ITC1/40.95
        TCELL2 = ITC2/40.95
        TCELL3 = ITC3/40.95
        RFC1 = 0.5*(IFC1+65536)
        RFC2 = 0.5*(IFC2+65536)
        RFC3 = 0.5*(IFC3+65536)
C
C Calculate volume flowrates
C
        VOLTOP=(0.4228E-6*IFT+0.401784E-2)*IFT+0.345835E-1
        VOLBOT=(0.145672E-6*IFB+0.275464E-2)*IFB+0.473506E-1
C
C Calculate stream compositions in mole fraction methanol
C
        DIEL=(.830545E-6*RFC1-.8703983E-1)*RFC1+2311.37
        FEEDCOMP=COMP(TCELL1,DIEL)
        DIEL=(.11361E-5*RFC2-.116636)*RFC2+3030.66
        BOTCOMP=COMP(TCELL2,DIEL) - 0.029
        DIEL=(.476448E-6*RFC3-.527498E-1)*RFC3+1482.64
        TOPCOMP=COMP(TCELL3,DIEL)
C
C Calculate densities, and flowrates
C
        DENSITY=1.0-.000361*TCELL1-(.1957+0.000604*TCELL1)*FEEDCOMP
        FMASSFEED=VOLFEED*DENSITY
        FMOLEFEED=1000*FMASSFEED/(18.02+14.02*FEEDCOMP)
C
        DENSITY=1.0-.000361*TCELL2-(.1957+0.000604*TCELL2)*BOTCOMP
        FMASSBOT=VOLBOT*DENSITY
        FMOLEBOT=1000*FMASSBOT/(18.02+14.02*BOTCOMP)
C
        DENSITY=1.0-.000361*TCELL3(.1957+0.000604*TCELL3)*TOPCOMP
        FMASSTOP=VOLTOP*DENSITY
        FMOLETOP=1000*FMASSTOP/(18.02+14.02*TOPCOMP)
C
        FMASSREF=VOLREF*DENSITY
        FMOLEREF=1000*FMASSREF/(18.02+14.02*TOPCOMP)
C
        RETURN
        END
```

```
          FUNCTION COMP(TEMP,DIEL)
C
C This function determines the mole fraction of methanol
C corresponding to a given temperature and dielectric constant.
C
        D = DIEL
        KOUNT = 0
        GO TO 10
C
20      A=(-.855089E-3*X+.150343E-2)*X+.00199439
        D=10**(LOG10(DIEL)-A*(25-TEMP))
10      X=((((.574151E-8*D-.16818E-5)*D+.190257E-3)*D
        1                       -.101264E-1)*D+.223373)*D-.421708
        KOUNT=KOUNT+1
        IF(KOUNT.LT.5) GO TO 20
C
        COMP = X
C
        RETURN
        END

          SUBROUTINE Control_out
C
C Send controller set points to microprocessor
C
        COMMON/ COL1 / TEMP1,TEMP8,TOPCOMP,BOTCOMP,FEEDCOMP
        COMMON/ COL2 / VOLTOP,VOLBOT,VOLREF,VOLFEED,STMFLOW
        COMMON/ COL3 / FMASSTOP,FMASSBOT,FMASSREF,FMASSFEED
        COMMON/ COL4 / FMOLETOP,FMOLEBOT,FMOLEREF,FMOLEFEED
C
        IFF=(-3.58183*VOLFEED+246.079)*VOLFEED+8.95361
        IRF=(-41.4991*VOLREF+360.049)*VOLREF-25.4994
        ISF=(324.886*STMFLOW-39.5401)*STMFLOW-129.679+0.5
C
        WRITE(6,100) IFF,IRF,ISF
100     FORMAT(' #CF',I4.4,' R',I4.4,' S',I3.3)
        RETURN
        END

          SUBROUTINE Esc_set
C
C Set AST to routine ESC_REC on receipt of an <ESC> character from
C the operator terminal.
C
        COMMON/ COL5 / TIME,ESCFLG,ICHAN
        LOGICAL ESCFLG
        DIMENSION MASK(2)
        EXTERNAL ESC_REC
C
        INCLUDE '($IODEF)'
C
        ESCFLG = .FALSE.
        MASK(1) = 0
        MASK(2) = '8000000'X
        CALL SYS$ASSIGN('TT',ICHAN,,)
        CALL SYS$QIOW(,%VAL(ICHAN),%VAL(IO$_SETMODE.OR.
        1           IO$M_OUTBAND),,,,ESC_REC,%REF(MASK),%VAL(4),,,)
C
        RETURN
        END
```

```
          SUBROUTINE Esc_rec
C
C AST service routine. Sets the ESCFLG flag on receipt of an
C <ESC> character.
C
        COMMON/ COL5 / TIME,ESCFLG,ICHAN
        LOGICAL ESCFLG
C
        ESCFLG = .TRUE.
C
        RETURN
        END

          SUBROUTINE Esc_clear
C
C Cancel AST's on receipt of <ESC>
C
        COMMON/ COL5 / TIME,ESCFLG,ICHAN
        LOGICAL ESCFLG
C
        CALL SYS$CANCEL (ICHAN)
C
        RETURN
        END

          SUBROUTINE Wakevery(S)
C
C Routine to schedule a hibernating process to wake every S
C seconds.  If S=0.0 no more wakeups will be scheduled and those
C pending will be cancelled
C
        REAL*4 S
        INTEGER*4 Ians(2)
C
        IF(ABS(S).LT.0.000001)THEN
           CALL SYS$CANWAK(,)
           RETURN
        ENDIF
C
C Calculate delta time in the system's 64bit clock format
C
        Itime = S*100.0
        Icons = -1000*100
        CALL LIB$EMUL(Itime,Icons,Izero,Ians(1))
C
        CALL SYS$SCHDWK(,,Ians(1),Ians(1))
        RETURN
        END
```

```
          SUBROUTINE Sleep
C
C Routine to hibernate the process until awoken by a scheduled
C wakeup
C
        CALL SYS$HIBER()
        RETURN
        END

          SUBROUTINE Wait(S)
C
C Routine to hibernate the process for S seconds
C
        REAL*4 S
        INTEGER*4 Ians(4)
C
C Calculate delta time in the system's 64bit clock format
C
        Itime = S*100.0
        Icons = -1000*100
        CALL LIB$EMUL(Itime,Icons,Izero,Ians(1))
C
        Ians(3)=0
        Ians(4)=0
        CALL SYS$SCHDWK(,,Ians(1),Ians(3))
        CALL SYS$HIBER()
        RETURN
        END
```

B.2 COMPOSITION ALGORITHM

COLUMN calculates liquid composition from dielectric constant and temperature using the following iterative procedure

1. Guess an initial value of the constant α, typically 2E-3.

2. Calculate the dielectric constant of the water/methanol mixture at 25°C using the following relation between temperature and the dielectric constant :

$$\log_{10}D^{\circ} = \log_{10}D - \alpha*(25-T)$$

where D is the dielectric constant at temperature T and D° is the dielectric constant at 25°C.

3. Calculate the liquid composition from the dielectric constant at 25°C by the equation :

$$x = 5.741E\text{-}9*D^{\circ 5} - 1.682E\text{-}6*D^{\circ 4} + 1.903E\text{-}4*D^{\circ 3} - 1.013E\text{-}2*D^{\circ 2} + 0.2234*D^{\circ} - 0.4217$$

4. Calculate a new value of α, which depends on the liquid composition according to :

$$\alpha = -8.551E\text{-}4*x^2 + 1.503E\text{-}3*x + 1.994E\text{-}3$$

5. Compare the new value of α with the previous value and if necessary loop back to step 2.

APPENDIX C

SENSOR CALIBRATIONS

C.1 STEAM FLOWRATE

$$D = 324.9*S^2 - 39.54*S - 129.7$$

where S is the steam condensate flowrate in $kg.min^{-1}$ and D is the output from the microprocessor in D/A converter units.

C.2 TURBINE FLOWMETERS

Reflux flowrate sensor (No 1) :

$$Freq = -41.5*F^2 + 360*F - 25.5$$

Distillate flowrate sensor (No 2) :

$$F = 4.228E-7*Freq^2 + 4.018E-3*Freq + 3.458E-2$$

Bottoms flowrate sensor (No 3) :

$$F = 1.457E-7*Freq^2 + 2.755E-3*Freq + 4.735E-2$$

Feed flowrate sensor (No 4) :

$$Freq = -3.582*F^2 + 246.1*F + 8.954$$

where F is the flowrate in $l.min^{-1}$ and Freq is the output frequency in Hz.

C.3 CAPACITANCE CELLS

Feed cell :

$$Diel = 8.305E\text{-}7*Freq^2 - 8.704E\text{-}2*Freq + 2311.4$$

Bottoms cell :

$$Diel = 1.136E\text{-}6*Freq^2 - 0.1166*Freq + 3030.7$$

Distillate cell :

$$Diel = 4.765E\text{-}7*Freq2 - 0.5275*Freq + 1482.6$$

where Diel is the dielectric constant and Freq is the oscillator frequency in Hz.

APPENDIX D

DYNAMIC COLUMN SIMULATION

D.1 Program Listing

```
        PROGRAM Dynamic_Simulation
C
C A dynamic distillation column simulation program written
C by P.W.M Janssen
C
C
        INCLUDE 'COLCOM.FOR'
C
        CALL LIB$ERASE_PAGE (1,1)
        TYPE 100
100     FORMAT(/15X,'A Dynamic Binary Distillation',
        1       ' Program'/15x,37('-')/)
C
C Read system properties
C
        CALL RDPROP
C
C Read column parameters
C
        CALL RDPARA
C
C Fit cubic spline to eqilibrium data
C
        CALL EQSPLINE
C
C Calculate steady state solution
C
        CALL SSEVAL
C
C Calculate dynamic response
C
        CALL DYNRES
C
        END
```

```
C
C Common block for dynamic column simulation in file COLCOM.FOR
C
      IMPLICIT REAL*8 (A-H,L,O-Z)
      COMMON /COL1  / L(0:20),V(0:20),X(0:20),Y(0:20),VM(0:20)
      COMMON /COL2  / TT,TB,TF,TR,QS,QV,LL0,FL,F,D,XF
      COMMON /COL3  / EL(0:20),EV(0:20),DHX(20),ELF
      COMMON /COL4  / EQS(41),EQK(0:20),EFF(0:20),EMV(5)
      COMMON /COL5  / BAND(7,40),BVEC(40),IBPVT(40)
      COMMON /COL6  / Y0(40),RWORK(1982),IWORK(60),TIME,DELT
      COMMON /COL7  / QPLOSS,QRLOSS
      COMMON /COL8  / AREA1,AREA2,HGT1,CONPR,REBVOL
      COMMON /COL9  / TAREA,HWEIR,LWEIR,CAVITY
      COMMON /COL10 / RMSTL,RMSTD,TP(21)
      COMMON /COL11 / NP,NP1,NP2,NF,NDYN
      COMMON /COL12 / CP1,CP2,CP3,CP4,EL0,EL1,EL2,EL3,EL4,EL5
      COMMON /COL13 / EV0,EV1,EV2,EV3,EV4,EV5,AMW1,AMW2
      COMMON /COL14 / RH1,RH2,RH3,RH4,RH5,RH6,DHV
      COMMON /COL15 / TS0,TS1,TS2,TS3,TS4,TS5
      COMMON /COL16 / ACON(38),BCON(38),CCON(38),DCON(38)
      CHARACTER*80 FILENAME
C
```

```
        SUBROUTINE RDPROP
C
C Purpose: Input system properties from data file
C
        INCLUDE 'COLCOM.FOR'
        CHARACTER*80 HEADER
C
        WRITE(6,100)
100     FORMAT('$Enter physical properties data file name - ')
        ACCEPT 275, FILENAME
275     FORMAT(A)
        OPEN(UNIT=10,FILE=FILENAME,STATUS='OLD',ERR=100)
C
        READ(10,275) HEADER
        TYPE *,HEADER
C
        READ(10,310)CP1,CP2,CP3,CP4
        READ(10,310)EL0,EL1,EL2,EL3,EL4,EL5
        READ(10,310)EV0,EV1,EV2,EV3,EV4,EV5
        READ(10,310)AMW1,AMW2
        READ(10,310)RH1,RH2,RH3,RH4,RH5,RH6
        READ(10,310)TS0,TS1,TS2,TS3,TS4,TS5
        READ(10,310)DHV
        READ(10,350) (EQS(J),J=1,41)
C
        CALL CLOSE (10)
310     FORMAT(6E12.5)
350     FORMAT(9F7.4)
        TYPE *,'System data loaded'
C
        RETURN
        END
```

```
      SUBROUTINE RDPARA
C
C Purpose: To input column parameters from data file
C
      INCLUDE 'COLCOM.FOR'
C
      WRITE(6,100)
100   FORMAT(/'$Enter column parameters data file name - ')
      ACCEPT 275, FILENAME
275   FORMAT(A)
      OPEN(UNIT=10,FILE=FILENAME,STATUS='OLD',ERR=100)
C
      READ(10,310) NP
      READ(10,310) NF
      NP1=NP+1
      NP2=NP+2
      READ(10,350) TF,TR,TT,TB
      READ(10,350) (EMV(I),I=1,5)
      READ(10,350) QPLOSS,QRLOSS
      QPLOSS = QPLOSS*6D-2
      QRLOSS = QRLOSS*6D-2
      READ(10,350) REBVOL
      READ(10,350) AREA1,AREA2,HGT1,CONPR
      READ(10,350) TAREA,HWEIR,LWEIR,CAVITY
C
      CALL CLOSE (10)
310   FORMAT(5I10)
350   FORMAT(5F15.0)
      TYPE *,'Column parameters data loaded'
      RETURN
      END
```

```
        SUBROUTINE SSEVAL
C
C Purpose: To determine steady state operating conditions for
C distillation column simulation
C
        INCLUDE 'COLCOM.FOR'
C
        EXTERNAL DIFFUN,PEDREV
C
        WRITE(6,100)
100     FORMAT(/'$Enter column input file name - ')
        ACCEPT 275, FILENAME
275     FORMAT(A)
        OPEN(UNIT=11,FILE=FILENAME,STATUS='OLD')
C
C Set up initial compositions
C
        CALL INITSS
        IF(IFLAG1.NE.0) STOP ' Flow data inconsistent'
C
C Set up parameters for LSODE routine
C
        T0=0.0D0
        DO 35 I=1,NP2
35      Y0(I) = X(I-1)
        H0 = 1.0D-4
        ITOL = 1
        RTOL = 1.0D-4
        ATOL = 1.0D-6
        ITASK = 1
        ISTATE = 1
        IOPT = 0
        MF=22
        TOUT = 1.0D0
C
20      CALL LSODE(DIFFUN,NP2,Y0,T0,TOUT,ITOL,RTOL,ATOL,ITASK,
        1             ISTATE,IOPT,RWORK,1982,IWORK,60,PEDREV,MF)
C
        DO 967 I=0,NP1
967     X(I)=Y0(I+1)
C
C Check for convergence
C
        CALL CHKCON
        IF(RMSTD.LT.1D-6) GO TO 40
C
C Double integration length and try again
C
        TOUT = T0*2.0D0
        GO TO 20
C
40      TYPE 210
45      TYPE 215
210     FORMAT(/' Steady state solution reached'/
        1       ' 1 Display solution on terminal'/
        2       ' 2 Continue to dynamic simulation')
215     FORMAT('$Next command ? ')
C
        READ(5,220) IQR
220     FORMAT(A2)
```

```
      IF(IQR.EQ.1) GO TO 51
      IF(IQR.EQ.2) RETURN
      GO TO 40
C
C Display steady state solution on terminal
C
51    D=V(1)-L(0)
      WRITE(6,230)
      WRITE(6,240)
      WRITE(6,250) (I,Y(I),X(I),V(I),L(I),I=1,NP1)
      WRITE(6,260) F,XF,ELF,D,X(0),L(0),L(NP1),X(NP1),EL(0)
      WRITE(6,120)
      WRITE(6,130) (I,EFF(I),EQK(I),EV(I),EL(I),I=1,NP1)
      GO TO 45
C
230   FORMAT(//' Steady state solution:'/)
240   FORMAT(' PLATE NO',5X,'VAP.COMP',5X,'LIQ.COMP',5X,
     1      'VAP.FLOW',5X,'LIQ.FLOW')
250   FORMAT(' ',I6,3X,4F13.6)
260   FORMAT(/' FEED FLOW - ',F10.6,'   FEED COMP - ',F10.7
     1      ,' FEED ENTH - ',F10.6
     2      /' TOPS FLOW - ',F10.6,'   TOPS COMP - ',F10.7
     3      ,' REF  FLOW - ',F10.6
     4      /' BOTS FLOW - ',F10.6,'   BOTS COMP - ',F10.7
     5      ,' REF  ENTH - ',F10.6)
120   FORMAT(/' PLATE NO',5X,'MUR.EFF ',5X,'EQIL.VAL',5X,
     1       'VAP.ENTH',5X,'LIQ.ENTH')
130   FORMAT(' ',I6,3X,4F13.6)
      END
```

```
      SUBROUTINE DYNRES
C
C Purpose: To calculate dynamic response of a distillation
C column using numerical  integration
C
      INCLUDE 'COLCOM.FOR'
C
      LOGICAL IFIRST
C
      EXTERNAL DIFFN2,PEDREV
C
C Open output data file
C
99    WRITE(6,101)
101   FORMAT(/'$Enter output data file name - ')
      ACCEPT 275, FILENAME
275   FORMAT(A)
      OPEN(UNIT=12,FILE=FILENAME,STATUS='NEW',ERR=99)
C
C Write initial steady state values to output file
C
      CALL FILOUT
C
C Find column find tray molar holdups, initialise time
C and initialise state vector.
C
      CALL MOLES
      TIME = 0.0D0
      DELT = 4.0D0
      NDYN = 2*NP1
      DO 35 I=0,NP
      J = 2*(I+1)-1
35    Y0(J) = X(I)
      Y0(NDYN) = X(NP1)
      DO 36 I=1,NP
36    Y0(2*I) = VM(I)
C
C Set integration parameters
C
      ITOL = 1
      RTOL = 1.0D-8
      ATOL = 1.0D-8
      ITASK = 1
      IOPT = 0
      MF=22
      IFIRST = .TRUE.
C
C Numerical  integration in time domain loop starts here
C
10    CONTINUE
C
C Read new column inputs
C
      READ(11,823,END=30)FNEW,LNEW,QNEW,XNEW
823   FORMAT(4F10.0)
C
      IF(FL.NE.FNEW.OR.LL0.NE.LNEW.OR.QS.NE.QNEW.OR.XNEW.NE.XF
     1       .OR.IFIRST) THEN
```

```
C
C Set column inputs to new values
C
            FL = FNEW
            LL0 = LNEW
            QS = QNEW
            XF = XNEW
C
C Convert data units
C
            RHO=RH1+RH2*TF+(RH3+RH4*TF)*XF
            F=FL*RHO*1000.0/(AMW1*XF+AMW2)
            QV=QS*DHV-QRLOSS
            ELF=ENSUB(XF,TF)
C
C Set integration parameters for new start
C
               ISTATE = 1
         ELSE
               ISTATE = 2
         END IF
C
C Write results to output file
C
         CALL FILOUT
C
         TOUT = TOUT + DELT
C
         CALL LSODE(DIFFN2,NDYN,Y0,TIME,TOUT,ITOL,RTOL,ATOL,
         1       ITASK,ISTATE,IOPT,RWORK,1982,IWORK,60,PEDREV,MF)
C
         IFIRST = .FALSE.
C
         GO TO 10
C
30       CONTINUE
C
C Write last results to output file
C
         CALL FILOUT
C
         CALL CLOSE(12)
         WRITE(6,100) TIME
100      FORMAT(' Numerical  integration complete until -',
         1                                          F7.2,'mins')
         RETURN
         END
```

```
      SUBROUTINE INITSS
C
C Purpose: To initialise compositions and flows for the steady
C state program "SSEVAL"
C
      INCLUDE 'COLCOM.FOR'
C
C Read initial column inputs
C
      READ(11,823)FL,LL0,QS,XF
823   FORMAT(4F10.0)
C
C Convert data units
C
      RHO=RH1+RH2*TF+(RH3+RH4*TF)*XF
      F=FL*RHO*1000.0/(AMW1*XF+AMW2)
      QV=QS*DHV-QRLOSS
      ELF=ENSUB(XF,TF)
C
C Set reboiler efficiency at unity
C
      EFF(NP1)=1.0
C
C Check initial temp guesses
C
      TOP=TS5+TS4+TS3+TS2+TS1+TS0
      BOT=TS0
      DT=BOT-TOP
      TOP=TOP+0.05*DT
      BOT=BOT-0.05*DT
      IF(TT.LT.TOP)TT=TOP
      IF(TB.GT.BOT)TB=BOT
C
C Assume feed plate comp = XF
C
      TNF=((((TS5*XF+TS4)*XF+TS3)*XF+TS2)*XF+TS1)*XF+TS0
      IF(TT.GT.TNF)TT=TNF
      IF(TB.LT.TNF)TB=TNF
C
C Set up dual linear temp profile and find liquid compositions
C
      RN=NF
      DO I=0,NFM1
         RI=I/RN
         TP(I)=(1.0-RI)*TT+RI*TNF
         X(I)=COMP(TP(I))
      END DO
C
      X(NF)=XF
      TP(NF)=TNF
      RN=NP1-NF
C
      DO I=NF1,NP1
         RI=(I-NF)/RN
         TP(I)=(1.0-RI)*TNF+RI*TB
         X(I)=COMP(TP(I))
      END DO
      TP(21)=TP(1)
```

```
C
        TYPE *,'Initial temperatures & compositions'
        WRITE(6,100) (TP(J),X(J),J=1,NP1)
100     FORMAT(2F15.7)
C
        RETURN
        END


        SUBROUTINE CHKCON
C
C Purpose: To check convergence of steady state solution
C
        INCLUDE 'COLCOM.FOR'
C
C Find sum of temp errors squared SES
C
        SES=0.0
        DO I=1,NP1
           XT=X(I)
           TI=((((TS5*XT+TS4)*XT+TS3)*XT+TS2)*XT+TS1)*XT+TS0
           DIF=TP(I)-TI
           SES=SES+DIF*DIF
           TP(I)=TI
        END DO
C
        RMSTD=SQRT(SES)/(NP1)
        RETURN
        END
```

```
      SUBROUTINE DIFFUN (IDUM,TDUM,YVAL,YDER)
C
C Purpose: To calculate the derivatives of the liquid
C compositions. Vapour compositions, and column flows are
C calculated as intermediate results.
C
      INCLUDE 'COLCOM.FOR'
C
      DIMENSION YVAL(NP2),YDER(NP2)
C
C Set liquid compositions
C
      DO 5 I=0,NP1
5     X(I) = YVAL(I+1)
C
C Calculate equilibrium constants and murfree efficiencies
C
      CALL EQILIB
      CALL MUREFF
C
C Calculate vapour compositions
C
      Y(NP1)=EQK(NP1)*X(NP1)
      DO 10 I=NP,1,-1
10    Y(I)=EQK(I)*EFF(I)*X(I)-(EFF(I)-1)*Y(I+1)
C
C Calculate liquid & vapour enthalpies
C
      CALL ENTHPY
C
C Change reflux flow to molar units
C
      RHO=RH1+RH2*TR+(RH3+RH4*TR)*X(0)
      L(0)=LL0*RHO*1000.0/(AMW1*X(0)+AMW2)
C
C Set up banded matrix
C
      DO J=1,NP1
         I=2*J-1
         BAND(7,I)   = 0.0D0
         BAND(7,I+1) = -1.0D0
         BAND(6,I)   = 1.0D0
         BAND(6,I+1) = -1.0D0*EL(J)
         BAND(5,I)   = EV(J)
         BAND(5,I+1) = 1.0D0
         BAND(4,I)   = -1.0D0
         BAND(4,I+1) = EL(J)
         BAND(3,I)   = -1.0D0*EV(J)
         BAND(3,I+1) = 0.0D0
         BAND(2,I)   = 0.0D0
         BAND(2,I+1) = 0.0D0
         BAND(1,I)   = 0.0D0
         BAND(1,I+1) = 0.0D0
         BVEC(I)     = -1.0D0*QPLOSS
         BVEC(I+1)   = 0.0D0
      END DO
```

```
C
C Set up B vector
C
        N2NP1 = 2*NP1
        BVEC(1)        = BVEC(1)+L(0)*EL(0)
        BVEC(2)        = L(0)
        BVEC(2*NF-1)   = BVEC(2*NF-1)+F*ELF
        BVEC(2*NF)     = F
        BVEC(N2NP1-1) = QV
C
C Call LU decomposition routine for banded matrices
C
        CALL DGBFA(BAND,7,N2NP1,2,2,IBPVT,ISTAT)
        IF (ISTAT .NE. 0) STOP 'No solution to banded matrix'
C
C Call solution routine for banded matrices
C
        CALL DGBSL(BAND,7,N2NP1,2,2,IBPVT,BVEC,0)
C
C Evaluate column flows
C
        DO 30 I=1,NP1
        J = 2*I-1
        V(I) = BVEC(J)
30      L(I) = BVEC(J+1)
C
C Calculate derivatives of liquid composiotns
C
        YDER(1) = V(1)*(Y(1)-X(0))
        DO 40 I=1,NP
        YDER(I+1) = (L(I-1)*X(I-1)+V(I+1)*Y(I+1)-L(I)*X(I)
        1       -V(I)*Y(I))
40      CONTINUE
        YDER(NF+1) = YDER(NF+1)+F*XF
        YDER(NP2) = (L(NP)*(X(NP)-X(NP1))+V(NP1)*(X(NP1)-Y(NP1)))
C
        RETURN
        END


        SUBROUTINE PEDREV(N,T,Y,ID1,ID2,PW,M)
C
        TYPE *,'Jacobian function not implemented'
C
        RETURN
        END
```

```
      SUBROUTINE DIFFN2 (IDUM,TDUM,YVAL,YDER)
C
C Purpose: To calculate the derivatives of the liquid compositions
C and molar holdups
C
      INCLUDE 'COLCOM.FOR'
C
      DIMENSION YVAL(NDYN),YDER(NDYN)
C
C Set liquid compositions
C
      DO 5 I=0,NP
      J=2*(I+1)-1
5     X(I) = YVAL(J)
      X(NP1) = YVAL(NDYN)
C
C Set molar volumes
C
      DO 6 I=1,NP
6     VM(I) = YVAL(2*I)
C
C Calculate condenser molar holdup
C
      DENMAS=RH5+RH6*X(0)
      HGT2=CONPR/(DENMAS*9.8D3)-HGT1
      VM(0)=(HGT1*AREA1+HGT2*AREA2)*DENSTY(X(0))*1000.0
C
C Calculate reboiler molar holdup
C
      VM(NP1)=REBVOL*DENSTY(X(NP1))
C
C Calculate equilibrium constants, murfree vapour efficiencies
C and dh/dX terms
C
      CALL EQILIB
      CALL MUREFF
      CALL DHDX
C
C Calculate vapour compositions
C
      Y(NP1)=EQK(NP1)*X(NP1)
      DO 10 I=NP,1,-1
10    Y(I)=EQK(I)*EFF(I)*X(I)-(EFF(I)-1)*Y(I+1)
C
C Calculate liquid & vapour enthalpies
C
      CALL ENTHPY
C
C Evaluate liquid flows
C
      RHO=RH1+RH2*TR+(RH3+RH4*TR)*X(0)
      L(0)=LL0*RHO*1D3/(AMW1*X(0)+AMW2)
      DO 30 I=1,NP
      HL=(VM(I)/(DENSTY(X(I))*1D3)-CAVITY)/TAREA
      HOW=(HL/0.65D0)-HWEIR
      IF (HOW .LE. 0D0) THEN
            TYPE *,'HEIGHT =< 0  ON TRAY ',I
            L(I) = 0D0
      ELSE
            Q=LWEIR*(HOW/4.43D-4)**1.5D0
```

```
                L(I)=DENSTY(X(I))*Q
        END IF
30      CONTINUE
C
C Evaluate vapour flows
C
        V(NP1)=(QV+L(NP)*(EL(NP)-EL(NP1)-DHX(NP1)*(X(NP)
        1 -X(NP1))))/(EV(NP1)-EL(NP1)-DHX(NP1)*(Y(NP1)-X(NP1)))
        DO 35 I=NP,1,-1
        V(I)=(V(I+1)*(EV(I+1)-EL(I)-DHX(I)*(Y(I+1)-X(I)))
        1  +L(I-1)*(EL(I-1)-EL(I)-DHX(I)*(X(I-1)-X(I)))-QPLOSS)
        2        /(EV(I)-EL(I)-DHX(I)*(Y(I)-X(I)))
        IF (I .NE. NF) GO TO 35
        V(I)=V(I)+(F*(ELF-EL(I)-DHX(I)*(XF-X(I))))
        1        /(EV(I)-EL(I)-DHX(I)*(Y(I)-X(I)))
35      CONTINUE
C
C Calculate derivatives of liquid compositions
C
        YDER(1) = (V(1)*(Y(1)-X(0)))/VM(0)
        DO 40 I=1,NP
        YDER(2*I+1) = (L(I-1)*(X(I-1)-X(I))+V(I+1)*(Y(I+1)-X(I))
        1        -V(I)*(Y(I)-X(I)))/VM(I)
40      CONTINUE
        YDER(2*NF+1) = YDER(2*NF+1)+F*(XF-X(NF))/VM(NF)
        YDER(NDYN) = (L(NP)*(X(NP)-X(NP1))-V(NP1)*(Y(NP1)
        1                                        -X(NP1)))/VM(NP1)
C
C Calculate derivatives of molar holdups
C
        DO 50 I=1,NP
        YDER(2*I) = V(I+1)+L(I-1)-L(I)-V(I)
50      CONTINUE
        YDER(2*NF) = YDER(2*NF)+F
C
        RETURN
        END


        SUBROUTINE DHDX
C
C Purpose: To calculate dh/dX as a function of X, for all plates
C
        INCLUDE 'COLCOM.FOR'
C
        DO I=0,NP1
           DHX(I)=(((EL5*5D0*X(I)+EL4*4D0)*X(I)+EL3*3D0)
        1                *X(I)+EL2*2D0)*X(I)+EL1
        END DO
        RETURN
        END
```

```
      SUBROUTINE ENTHPY
C
C Purpose: To find liquid and vapour enthalpies for each plate
C
      INCLUDE 'COLCOM.FOR'
C
      DO I=1,NP1
         YT=Y(I)
         EL(I)=ENSAT(X(I))
         EV(I)=((((EV5*YT+EV4)*YT+EV3)*YT+EV2)*YT+EV1)*YT+EV0
      END DO
C
C Calculate reflux enthalpy
C
      EL(0)=ENSUB(X(0),TR)
C
      RETURN
      END


      FUNCTION ENSUB(COM,TEMP)
C
C Purpose: To find enthalpy of subcooled liquid
C
      INCLUDE 'COLCOM.FOR'
C
      ENSAT=((((EL5*COM+EL4)*COM+EL3)*COM+EL2)*COM+EL1)*COM+EL0
      SATEMP=((((TS5*COM+TS4)*COM+TS3)*COM+TS2)*COM+TS1)*COM+TS0
C
C Check if saturated
C
      IF(TEMP.GE.SATEMP) THEN
         ENSUB=ENSAT
      ELSE
         HCPSAT=(CP1*SATEMP+CP2)*COM+CP3*SATEMP+CP4
         HCPSUB=(CP1*TEMP+CP2)*COM+CP3*TEMP+CP4
         ENSUB=ENSAT-((HCPSAT+HCPSUB)/2.0)*(SATEMP-TEMP)
      END IF
C
      RETURN
      END
```

```
      FUNCTION ENSAT(XX)
C
C Purpose: To evaluate saturated liquid enthalpy
C
      INCLUDE 'COLCOM.FOR'
C
      ENSAT=((((EL5*XX+EL4)*XX+EL3)*XX+EL2)*XX+EL1)*XX+EL0
C
      RETURN
      END


      SUBROUTINE EQILIB
C
C Purpose: To calculate a "k" value for each plate
C
      INCLUDE 'COLCOM.FOR'
C
      DO I=1,NP1
         IJ=JIDINT(X(I)*4D1)
         IF (IJ .LE. 0) IJ = 1
         IF (IJ .GE.39) IJ = 38
         EQK(I) = (((ACON(IJ)*X(I)+BCON(IJ))*X(I)
     1               +CCON(IJ))*X(I)+DCON(IJ))/X(I)
      END DO
C
      RETURN
      END


      SUBROUTINE EQSPLINE
C
C Purpose: To calculate a cubic spline to fit the liquid/vapour
C equilibrium data
C
      INCLUDE 'COLCOM.FOR'
C
      DIMENSION BND(13,160),BV(160),IPV(160)
C
C Set up banded matrix
C
      DO I=1,37
         I1=I*4
         I2=I*4+1
         I3=I*4+2
         I4=I*4+3
         FI=I
         XTEMP1=(FI+1D0)/4D1
         XTEMP2=(FI+2D0)/4D1
         BND(13,I1) = 1D0
         BND(11,I1) = XTEMP2
         BND(11,I3) = 2D0*XTEMP2
         BND(10,I1) = XTEMP1
         BND(10,I2) = 1D0
         BND(10,I3) = 2D0
         BND(10,I4) = 3D0*XTEMP2*XTEMP2
         BND(9,I1) = -1D0
         BND(9,I2) = 1D0
         BND(9,I3) = XTEMP2*XTEMP2
         BND(9,I4) = 6D0*XTEMP2
```

```
            BND(8,I3) = XTEMP1*XTEMP1
            BND(8,I4) = XTEMP2*XTEMP2*XTEMP2
            BND(7,I3) = -2D0*XTEMP1*XTEMP1
            BND(7,I4) = XTEMP1*XTEMP1*XTEMP1
            BND(6,I3) = -2D0
            BND(6,I4) = -3D0*XTEMP1*XTEMP1
            BND(5,I4) = -6D0*XTEMP1
            BV(I2)    = EQS(I+2)
            BV(I3)    = EQS(I+3)
         END DO
C
         XTEMP1= 1D0/4D1
         XTEMP2= 2D0/4D1
         BND(12,1) = 1D0
         BND(10,1) = XTEMP2
         BND(9,1) = XTEMP1
         BND(11,2) = 2D0*XTEMP2
         BND(10,2) = 2D0
         BND(9,2) = XTEMP2*XTEMP2
         BND(8,2) = XTEMP1*XTEMP1
         BND(10,3) = 3D0*XTEMP2*XTEMP2
         BND(9,3) = 6D0*XTEMP2
         BND(8,3) = XTEMP2*XTEMP2*XTEMP2
         BND(7,3) = XTEMP1*XTEMP1*XTEMP1
         BV(1)     = EQS(2)
         BV(2)     = EQS(3)
C
         BND(12,148) = 1D0
         BND(11,149) = 1D0
         BND(10,150) = 1D0
         BND(9,151)  = 1D0
         BV(151)     = 1D0
C
C Solve banded linear equations using LINPACK routines
C
         CALL DGBFA(BND,13,151,4,4,IPV,ISTAT)
         CALL DGBSL(BND,13,151,4,4,IPV,BV,0)
C
C Evaluate cubic spline coefficients
C
         ACON(1)=BV(3)
         BCON(1)=BV(2)
         CCON(1)=BV(1)
         DCON(1)=0
C
         DO I=2,38
            J=I*4-4
            ACON(I)=BV(J+3)
            BCON(I)=BV(J+2)
            CCON(I)=BV(J)
            DCON(I)=BV(J+1)
         END DO
C
         RETURN
         END
```

```
      SUBROUTINE MUREFF
C
C Purpose: To calculate a murphree vapour efficiency for each
C plate
C
      INCLUDE 'COLCOM.FOR'
C
      DO I=1,NP
         XT=X(I)
         EFF(I)=(((EMV(5)*XT+EMV(4))*XT+EMV(3))*XT+EMV(2))*XT
     1                                                  +EMV(1)
         IF(EFF(I).GT.1.00) EFF(I)=1.00
         IF(EFF(I).LT.0.01) EFF(I)=0.01
      END DO
C
      RETURN
      END


      FUNCTION COMP(TEMP)
C
C Purpose: To find liquid composition as a function of saturated
C temperature using secant method
C
      INCLUDE 'COLCOM.FOR'
C
      SUMT=TS0+TS1+TS2+TS3+TS4+TS5
      X1=0
      GX1=TS0-TEMP
      AGX1=ABS(GX1)
      X2=1
      GX2=SUMT-TEMP
      AGX2=ABS(GX2)
C
      DO 25 K=1,50
         X3=X1+GX1*(X2-X1)/(GX1-GX2)
         GX3=((((TS5*X3+TS4)*X3+TS3)*X3+TS2)*X3+TS1)*X3+TS0)
     1                                                  -TEMP
         AGX3=ABS(GX3)
         IF(AGX3.LT.0.0001) GO TO 45
         IF(AGX1.LT.AGX2) GO TO 15
         X1=X3
         GX1=GX3
         AGX1=AGX3
         GO TO 25
15       X2=X3
         GX2=GX3
         AGX2=AGX3
25    CONTINUE
      TYPE *,'Compositions failed to converge'
      RETURN
45    COMP=X3
      RETURN
      END
```

```
      SUBROUTINE MOLES
C
C Purpose: To calculate tray molar holdups from steady state
C liquid flow data
C
      INCLUDE 'COLCOM.FOR'
C
      TCONS=2D0/3D0
      DO I=1,NP
         Q=L(I)/DENSTY(X(I))
         HOW=4.43D-4*(Q/LWEIR)**TCONS
         HL=0.65D0*(HOW+HWEIR)
         VM(I)=(HL*TAREA+CAVITY)*DENSTY(X(I))*1D3
      END DO
C
C Calculate condenser molar holdup
C
      XTEMP = X(0)
      DENMAS=RH5+RH6*XTEMP
      HGT2=CONPR/(DENMAS*9.8D3)-HGT1
      VM(0)=(HGT1*AREA1+HGT2*AREA2)*DENSTY(XTEMP)*1000.0
C
C Calculate reboiler molar holdup
C
      XTEMP = X(NP1)
      VM(NP1)=REBVOL*DENSTY(XTEMP)
C
      TYPE *,'In Moles Routine'
      TYPE *,(VM(I),I=0,NP1)
C
      RETURN
      END


      FUNCTION DENSTY(XX)
C
C Purpose: To find the density of saturated liquid of
C composition xx in moles/litre
C
      INCLUDE 'COLCOM.FOR'
C
      DENSTY=(RH5+RH6*XX)*1000.0/(AMW1*XX+AMW2)
      RETURN
      END
```

D.2 PROPERTIES DATA FILE

```
Methanol/Water binary system data
0.6700E-07 0.1260E-01 0.1340E-04 0.7467E-01
0.7524E+01-0.1363E+02 0.3393E+02-0.3897E+02 0.1647E+02 0.0000E+00
0.4821E+02-0.9084E+01 0.1490E+01 0.0000E+00 0.0000E+00 0.0000E+00
0.14024E+02 0.18016E+02
0.10065E+01-0.36162E-03-0.19574E+00-0.60352E-03 0.97000E+00
-0.22230E+00
0.9993E+02-0.1627E+03 0.5019E+03-0.8742E+03 0.7398E+03-0.2400E+03
0.22800E+04
0.0000 0.1550 0.2670 0.3450 0.4180 0.4700 0.5170 0.5480 0.5790
0.6060 0.6270 0.6470 0.6650 0.6820 0.6970 0.7120 0.7290 0.7380
0.7500 0.7640 0.7790 0.7880 0.8000 0.8130 0.8250 0.8310 0.8460
0.8580 0.8700 0.8820 0.8930 0.9040 0.9150 0.9260 0.9360 0.9470
0.9580 0.9670 0.9790 0.9900 1.0000
```

D.3 PARAMETERS DATA FILE

```
8
5
20,20,70,80
.614,.786,-1.03,.436,0.0
189,1500
30.0
9.6E-4,4.0E-2,.230,4.24E3
.0316,.019,.168,.16E-3
```

APPENDIX E

METHANOL/WATER PROPERTIES

E.1 DIELECTRIC CONSTANTS

Mole fraction methanol	Temperature [deg C]			
	5	15	25	35
0.0000	87.74	81.95	78.30	74.83
0.0588	81.68	77.83	74.18	70.68
0.1233	77.38	73.59	69.99	66.52
0.1942	72.80	69.05	65.55	62.20
0.2727	67.91	64.31	60.94	57.72
0.3600	62.96	59.54	56.28	53.21
0.4576	57.92	54.71	51.67	48.76
0.5675	52.96	49.97	47.11	44.42
0.6923	48.01	45.24	42.60	40.08
0.8350	42.90	40.33	37.91	35.65
0.9144	39.98	37.61	35.38	33.28
1.0000	36.88	34.70	32.66	30.74

This data comes from the paper of Akerlof (1932). He used a capacitance cell containing two 50mm diameter tantalum plates spaced 2mm apart and a capacitance bridge with a 2MHz oscillator to measure capacitance.

E.2 LIQUID/VAPOUR EQUILIBRIUM DATA

x	y	x	y
0.000	0.000	0.500	0.779
0.025	0.155	0.525	0.788
0.050	0.267	0.550	0.800
0.075	0.345	0.575	0.813
0.100	0.418	0.600	0.825
0.125	0.470	0.625	0.831
0.150	0.517	0.650	0.846
0.175	0.548	0.675	0.858
0.200	0.579	0.700	0.870
0.225	0.606	0.725	0.882
0.250	0.627	0.750	0.893
0.275	0.647	0.775	0.904
0.300	0.665	0.800	0.915
0.325	0.682	0.825	0.926
0.350	0.697	0.850	0.936
0.375	0.712	0.875	0.947
0.400	0.729	0.900	0.958
0.425	0.738	0.925	0.967
0.450	0.750	0.950	0.979
0.475	0.764	0.975	0.990
		1.000	1.000

where x and y are the equilibrium liquid and vapour compositions in mole fraction.

APPENDIX F

SIMULATED DATA

F.1 SIMULATED COLUMN OPERATING POINTS

To determine the validity region of the bilinear model, an input function was generated by stepping between the following operating points. Random binary signals with amplitudes in the range of 3 to 5% of the absolute value of the inputs were then superimposed on the step functions.

Operating Point No	Column Inputs		
	Feed [l/min]	Steam [kg/min]	Reflux [l/min]
1	1.750	1.100	0.950
2	1.750	1.100	0.842
3	1.750	0.988	0.842
4	1.558	0.988	0.842
5	1.558	0.988	0.746
6	1.558	0.887	0.746
7	1.388	0.887	0.746
8	1.388	0.887	0.660
9	1.388	0.796	0.660
10	1.235	0.796	0.660
11	1.235	0.796	0.585
12	1.235	0.715	0.585
13	1.100	0.715	0.585

Table F.1

F.2 GAINS FOR SIMULATED COLUMN

The gains for the simulated column at each of the thirteen operating points shown in Appendix F.1 were calculated using a sensitivity analysis and are shown below.

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	.022	.127	-.158	.438	.445	-1.16
2	.059	.176	-.238	.451	.417	-1.13
3	.009	.121	-.133	.353	.402	-1.05
4	.025	.144	-.179	.494	.501	-1.31
5	.063	.193	-.265	.515	.478	-1.28
6	.011	.136	-.149	.397	.453	-1.18
7	.028	.161	-.201	.555	.562	-1.47
8	.071	.217	-.298	.578	.536	-1.44
9	.012	.152	-.168	.444	.508	-1.33
10	.031	.181	-.225	.623	.633	-1.65
11	.089	.256	-.347	.633	.584	-1.59
12	.013	.169	-.187	.497	.571	-1.49
13	.036	.205	-.253	.694	.706	-1.85

Table F.2

F.3 IDENTIFIED MODEL GAINS

Models of different types were identified using input/output data generated by the simulated column and the input functions discribed in Appendix F.1 . The gains of the identified models are shown below and can be compared with those in Appendix F.2 .

4th Order Linear Model

Tops Response to			Bottoms Response to		
Feed	Reflux	Steam	Feed	Reflux	Steam
0.020	0.162	-0.186	0.492	0.533	-1.35

Table F.3

4th Order Diagonal Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.015	0.139	-0.158	0.414	0.433	-1.11
2	0.020	0.140	-0.155	0.454	0.472	-1.17
3	0.015	0.144	-0.164	0.391	0.412	-1.13
4	0.017	0.155	-0.175	0.465	0.485	-1.24
5	0.023	0.155	-0.171	0.509	0.529	-1.31
6	0.016	0.159	-0.181	0.437	0.461	-1.26
7	0.019	0.171	-0.194	0.519	0.542	-1.39
8	0.025	0.172	-0.189	0.569	0.591	-1.46
9	0.018	0.176	-0.200	0.488	0.515	-1.41
10	0.020	0.189	-0.214	0.580	0.606	-1.55
11	0.027	0.190	-0.210	0.634	0.660	-1.63
12	0.019	0.195	-0.222	0.545	0.574	-1.57
13	0.022	0.209	-0.236	0.646	0.675	-1.73

Table F.4

4th Order Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.015	0.141	-0.160	0.413	0.431	-1.11
2	0.021	0.139	-0.155	0.454	0.474	-1.17
3	0.015	0.144	-0.164	0.391	0.408	-1.12
4	0.017	0.156	-0.176	0.463	0.483	-1.24
5	0.023	0.154	-0.171	0.508	0.531	-1.31
6	0.016	0.159	-0.181	0.438	0.457	-1.26
7	0.019	0.172	-0.194	0.517	0.540	-1.38
8	0.025	0.170	-0.189	0.568	0.594	-1.46
9	0.018	0.176	-0.200	0.489	0.511	-1.41
10	0.021	0.189	-0.214	0.578	0.604	-1.55
11	0.028	0.187	-0.209	0.634	0.662	-1.63
12	0.019	0.193	-0.220	0.546	0.570	-1.57
13	0.022	0.208	-0.236	0.644	0.672	-1.73

Table F.5

F.4 IDENTIFIED MODELS

This appendix contains the identified models for the simulated column work described in Chapter 7 using the input function shown in Graph 7.1.

4th order linear models, corresponding to Graph 7.4 :

$$\begin{aligned} x_D(k) = {} & 0.887622 \times x_D(k-1) + 0.288671 \times x_D(k-2) - 0.108586 \times x_D(k-3) \\ & - 0.131171 \times x_D(k-4) + 0.001068 \times F(k-1) + 0.001546 \times F(k-2) \\ & - 0.000332 \times F(k-3) - 0.000303 \times F(k-4) + 0.026327 \times L_R(k-1) \\ & + 0.009311 \times L_R(k-2) - 0.012454 \times L_R(k-3) - 0.011214 \times L_R(k-4) \\ & - 0.026323 \times Q_S(k-1) - 0.011996 \times Q_S(k-2) + 0.011115 \times Q_S(k-3) \\ & + 0.013394 \times Q_S(k-4) + 0.059911 \end{aligned}$$

$$\begin{aligned} x_W(k) = {} & 0.952731 \times x_W(k-1) + 0.372002 \times x_W(k-2) - 0.021579 \times x_W(k-3) \\ & - 0.320287 \times x_W(k-1) + 0.035430 \times F(k-1) + 0.000357 \times F(k-2) \\ & - 0.014636 \times F(k-3) - 0.012578 \times F(k-4) + 0.034500 \times L_R(k-1) \\ & + 0.002922 \times L_R(k-2) - 0.014617 \times L_R(k-3) - 0.015341 \times L_R(k-4) \\ & - 0.092124 \times Q_S(k-1) - 0.004991 \times Q_S(k-2) + 0.039266 \times Q_S(k-3) \\ & + 0.036563 \times Q_S(k-4) + 0.003241 \end{aligned}$$

4th order diagonal bilinear models, corresponding to Graph 7.7 :

$$
\begin{aligned}
x_D(k) = {} & 0.664655 \times x_D(k-1) + 0.258597 \times x_D(k-2) + 0.021298 \times x_D(k-3) \\
& + 0.018112 \times x_D(k-4) - 0.007268 \times F(k-1) + 0.086740 \times F(k-2) \\
& + 0.061874 \times F(k-3) + 0.035706 \times F(k-4) - 0.023838 \times L_R(k-1) \\
& + 0.008201 \times L_R(k-2) - 0.002713 \times L_R(k-3) - 0.00613 \times L_R(k-4) \\
& - 0.166397 \times Q_S(k-1) - 0.055855 \times Q_S(k-2) + 0.022665 \times Q_S(k-3) \\
& + 0.000319 \times Q_S(k-4) + 0.009500 \times x_D(k-1)F(k-1) \\
& - 0.091255 \times x_D(k-2)F(k-2) - 0.066192 \times x_D(k-3)F(k-3) \\
& - 0.037875 \times x_D(k-4)F(k-4) + 0.053874 \times x_D(k-1)L_R(k-1) \\
& + 0.001518 \times x_D(k-2)L_R(k-2) - 0.006247 \times x_D(k-3)L_R(k-3) \\
& - 0.001128 \times x_D(k-4)L_R(k-4) + 0.151471 \times x_D(k-1)Q_S(k-1) \\
& + 0.046337 \times x_D(k-2)Q_S(k-2) - 0.017415 \times x_D(k-3)Q_S(k-3) \\
& + 0.007046 \times x_D(k-4)Q_S(k-4) + 0.036435
\end{aligned}
$$

$$
\begin{aligned}
x_W(k) = {} & 0.604598 \times x_W(k-1) + 0.354933 \times x_W(k-2) + 0.149848 \times x_W(k-3) \\
& - 0.104385 \times x_W(k-4) + 0.036107 \times F(k-1) + 0.020844 \times F(k-2) \\
& + 0.005134 \times F(k-3) - 0.001125 \times F(k-4) \quad 0.031082 \times L_R(k-1) \\
& + 0.019159 \times L_R(k-2) + 0.005798 \times L_R(k-3) - 0.001796 \times L_R(k-4) \\
& - 0.078897 \times Q_S(k-1) - 0.049454 \times Q_S(k-2) - 0.012941 \times Q_S(k-3) \\
& + 0.002490 \times Q_S(k-4) - 0.002119 \times x_W(k-1)F(k-1) \\
& - 0.026449 \times x_W(k-2)F(k-2) - 0.017370 \times x_W(k-3)F(k-3) \\
& - 0.016558 \times x_W(k-4)F(k-4) + 0.040681 \times x_W(k-1)L_R(k-1) \\
& + 0.009673 \times x_W(k-2)L_R(k-2) - 0.011839 \times x_W(k-3)L_R(k-3) \\
& - 0.015065 \times x_W(k-4)L_R(k-4) - 0.146817 \times x_W(k-1)Q_S(k-1) \\
& + 0.013011 \times x_W(k-2)Q_S(k-2) + 0.039676 \times x_W(k-3)Q_S(k-3) \\
& + 0.046547 \times x_W(k-4)Q_S(k-4) + 0.009964
\end{aligned}
$$

4th order bilinear models, corresponding to Graph 7.10 :

$$
\begin{aligned}
x_D(k) = {} & 0.540504 \times x_D(k-1) + 0.253454 \times x_D(k-2) + 0.097841 \times x_D(k-3) \\
& + 0.057381 \times x_D(k-4) + 0.032403 \times F(k-1) + 0.084633 \times F(k-2) \\
& + 0.024238 \times F(k-3) + 0.030611 \times F(k-4) + 0.072901 \times L_R(k-1) \\
& - 0.011021 \times L_R(k-2) - 0.057242 \times L_R(k-3) - 0.038766 \times L_R(k-4) \\
& - 0.102878 \times Q_S(k-1) - 0.076980 \times Q_S(k-2) - 0.013654 \times Q_S(k-3) \\
& - 0.020130 \times Q_S(k-4) - 0.028432 \times x_D(k-1)F(k-1) \\
& - 0.007786 \times x_D(k-1)F(k-2) + 0.027105 \times x_D(k-1)F(k-3) \\
& + 0.010761 \times x_D(k-1)F(k-4) - 0.035949 \times x_D(k-2)F(k-1) \\
& - 0.058232 \times x_D(k-2)F(k-2) - 0.028878 \times x_D(k-2)F(k-3) \\
& - 0.030154 \times x_D(k-2)F(k-4) + 0.008018 \times x_D(k-3)F(k-1) \\
& - 0.027797 \times x_D(k-3)F(k-2) - 0.018755 \times x_D(k-3)F(k-3) \\
& - 0.013239 \times x_D(k-3)F(k-4) + 0.023362 \times x_D(k-4)F(k-1) \\
& + 0.004561 \times x_D(k-4)F(k-2) - 0.005511 \times x_D(k-4)F(k-3) \\
& + 0.000231 \times x_D(k-4)F(k-4) + 0.068900 \times x_D(k-1)L_R(k-1) \\
& + 0.096434 \times x_D(k-1)L_R(k-2) + 0.102973 \times x_D(k-1)L_R(k-3) \\
& + 0.083357 \times x_D(k-1)L_R(k-4) - 0.015459 \times x_D(k-2)L_R(k-1) \\
& + 0.002537 \times x_D(k-2)L_R(k-2) + 0.013840 \times x_D(k-2)L_R(k-3) \\
& + 0.006049 \times x_D(k-2)L_R(k-4) - 0.048848 \times x_D(k-3)L_R(k-1) \\
& - 0.033946 \times x_D(k-3)L_R(k-2) - 0.025970 \times x_D(k-3)L_R(k-3) \\
& - 0.021082 \times x_D(k-3)L_R(k-4) - 0.054708 \times x_D(k-4)L_R(k-1) \\
& - 0.048405 \times x_D(k-4)L_R(k-2) - 0.039444 \times x_D(k-4)L_R(k-3) \\
& - 0.031764 \times x_D(k-4)L_R(k-4) + 0.088056 \times x_D(k-1)Q_S(k-1) \\
& + 0.076486 \times x_D(k-1)Q_S(k-2) + 0.047931 \times x_D(k-1)Q_S(k-3) \\
& + 0.054414 \times x_D(k-1)Q_S(k-4) - 0.004900 \times x_D(k-2)Q_S(k-1) \\
& - 0.008780 \times x_D(k-2)Q_S(k-2) - 0.023854 \times x_D(k-2)Q_S(k-3) \\
& - 0.025152 \times x_D(k-2)Q_S(k-4) - 0.004972 \times x_D(k-3)Q_S(k-1) \\
& - 0.003664 \times x_D(k-3)Q_S(k-2) - 0.010694 \times x_D(k-3)Q_S(k-3) \\
& - 0.014480 \times x_D(k-3)Q_S(k-4) + 0.004143 \times x_D(k-4)Q_S(k-1) \\
& + 0.010220 \times x_D(k-4)Q_S(k-2) + 0.009905 \times x_D(k-4)Q_S(k-3) \\
& + 0.012218 \times x_D(k-4)Q_S(k-4) + 0.048893
\end{aligned}
$$

$$
\begin{aligned}
x_W(k) = {} & 0.568388 \times x_W(k-1) + 0.352706 \times x_W(k-2) + 0.146924 \times x_W(k-3) \\
& - 0.063833 \times x_W(k-4) + 0.036383 \times F(k-1) + 0.020645 \times F(k-2) \\
& + 0.006436 \times F(k-3) - 0.001888 \times F(k-4) - 0.031449 \times L_R(k-1) \\
& + 0.019178 \times L_R(k-2) + 0.006453 \times L_R(k-3) - 0.002451 \times L_R(k-4) \\
& - 0.078378 \times Q_S(k-1) - 0.049483 \times Q_S(k-2) - 0.015055 \times Q_S(k-3) \\
& + 0.002689 \times Q_S(k-4) + 0.017773 \times x_W(k-1)F(k-1) \\
& - 0.020872 \times x_W(k-1)F(k-2) - 0.037748 \times x_W(k-1)F(k-3) \\
& - 0.005938 \times x_W(k-1)F(k-4) + 0.008344 \times x_W(k-2)F(k-1) \\
& - 0.001136 \times x_W(k-2)F(k-2) - 0.000296 \times x_W(k-2)F(k-3) \\
& + 0.011350 \times x_W(k-2)F(k-4) - 0.013725 \times x_W(k-3)F(k-1) \\
& + 0.011791 \times x_W(k-3)F(k-2) + 0.020539 \times x_W(k-3)F(k-3) \\
& + 0.002031 \times x_W(k-3)F(k-4) - 0.016675 \times x_W(k-4)F(k-1) \\
& - 0.014647 \times x_W(k-4)F(k-2) - 0.003180 \times x_W(k-4)F(k-3) \\
& - 0.022153 \times x_W(k-4)F(k-4) + 0.062464 \times x_W(k-1)L_R(k-1) \\
& + 0.057984 \times x_W(k-1)L_R(k-2) + 0.038772 \times x_W(k-1)L_R(k-3) \\
& + 0.020834 \times x_W(k-1)L_R(k-4) + 0.007748 \times x_W(k-2)L_R(k-1) \\
& + 0.001106 \times x_W(k-2)L_R(k-2) - 0.012419 \times x_W(k-2)L_R(k-3) \\
& - 0.013674 \times x_W(k-2)L_R(k-4) - 0.005038 \times x_W(k-3)L_R(k-1) \\
& - 0.011999 \times x_W(k-3)L_R(k-2) - 0.006899 \times x_W(k-3)L_R(k-3) \\
& - 0.001234 \times x_W(k-3)L_R(k-4) - 0.028274 \times x_W(k-4)L_R(k-1) \\
& - 0.036192 \times x_W(k-4)L_R(k-2) - 0.029867 \times x_W(k-4)L_R(k-3) \\
& - 0.019237 \times x_W(k-4)L_R(k-4) - 0.088649 \times x_W(k-1)Q_S(k-1) \\
& - 0.026203 \times x_W(k-1)Q_S(k-2) - 0.012998 \times x_W(k-1)Q_S(k-3) \\
& - 0.014618 \times x_W(k-1)Q_S(k-4) - 0.058209 \times x_W(k-2)Q_S(k-1) \\
& - 0.014764 \times x_W(k-2)Q_S(k-2) - 0.000661 \times x_W(k-2)Q_S(k-3) \\
& + 0.005563 \times x_W(k-2)Q_S(k-4) - 0.017439 \times x_W(k-3)Q_S(k-1) \\
& + 0.015428 \times x_W(k-3)Q_S(k-2) + 0.018075 \times x_W(k-3)Q_S(k-3) \\
& + 0.021263 \times x_W(k-3)Q_S(k-4) + 0.009031 \times x_W(k-4)Q_S(k-1) \\
& + 0.042169 \times x_W(k-4)Q_S(k-2) + 0.029904 \times x_W(k-4)Q_S(k-3) \\
& + 0.047414 \times x_W(k-4)Q_S(k-4) + 0.010173
\end{aligned}
$$

APPENDIX G

EXPERIMENTAL DATA

G.1 GAINS FOR IDENTIFIED MODELS

The following tables contain the gains of models identified using experimental column data generated using the input functions shown in Graph 8.1.

Gains of Linear Models

Model	Tops Response to			Bottoms Response to		
Order	Feed	Reflux	Steam	Feed	Reflux	Steam
2nd	0.027	0.189	-0.221	0.364	0.196	-0.893
3rd	0.028	0.189	-0.223	0.365	0.204	-0.900
4th	0.029	0.188	-0.224	0.365	0.202	-0.897

Table G.1

2nd Order Diagonal Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.024	0.185	-0.214	0.283	0.210	-0.73
2	0.040	0.164	-0.201	0.215	0.129	-0.51
3	0.020	0.177	-0.203	0.410	0.506	-1.37
4	0.032	0.218	-0.254	0.315	0.222	-0.80
5	0.043	0.203	-0.244	0.268	0.166	-0.65
6	0.021	0.216	-0.246	0.483	0.578	-1.59
7	0.034	0.271	-0.313	0.368	0.235	-0.90

Table G.2

3rd Order Diagonal Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.025	0.178	-0.209	0.254	0.185	-0.66
2	0.047	0.168	-0.212	0.235	0.169	-0.57
3	0.019	0.166	-0.192	0.393	0.299	-1.21
4	0.034	0.221	-0.262	0.312	0.227	-0.79
5	0.052	0.213	-0.264	0.298	0.215	-0.73
6	0.019	0.210	-0.240	0.482	0.366	-1.47
7	0.037	0.290	-0.339	0.376	0.272	-0.93

Table G.3

4th Order Diagonal Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.026	0.177	-0.211	0.249	0.172	-0.63
2	0.053	0.169	-0.224	0.254	0.189	-0.62
3	0.018	0.157	-0.183	0.380	0.199	-1.10
4	0.037	0.221	-0.268	0.315	0.222	-0.79
5	0.059	0.215	-0.278	0.319	0.235	-0.78
6	0.018	0.201	-0.230	0.464	0.247	-1.34
7	0.040	0.297	-0.351	0.378	0.275	-0.94

Table G.4

2nd Order Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.024	0.185	-0.214	0.325	0.238	-0.85
2	0.040	0.163	-0.200	0.227	0.119	-0.54
3	0.020	0.177	-0.203	0.376	0.575	-1.32
4	0.032	0.217	-0.253	0.344	0.234	-0.87
5	0.043	0.202	-0.243	0.280	0.157	-0.67
6	0.021	0.216	-0.246	0.443	0.650	-1.53
7	0.034	0.270	-0.311	0.404	0.237	-0.98

Table G.5

3rd Order Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.025	0.180	-0.213	0.312	0.231	-0.81
2	0.050	0.166	-0.217	0.258	0.156	-0.62
3	0.019	0.162	-0.188	0.340	0.416	-1.14
4	0.036	0.221	-0.265	0.359	0.253	-0.92
5	0.055	0.211	-0.269	0.321	0.200	-0.78
6	0.019	0.205	-0.234	0.413	0.494	-1.37
7	0.039	0.292	-0.343	0.439	0.283	-1.08

Table G.6

4th Order Bilinear Model

Operating	Tops Response to			Bottoms Response to		
Point	Feed	Reflux	Steam	Feed	Reflux	Steam
1	0.027	0.177	-0.213	0.305	0.195	-0.77
2	0.059	0.168	-0.234	0.309	0.190	-0.74
3	0.018	0.150	-0.176	0.314	0.227	-0.95
4	0.039	0.221	-0.272	0.376	0.238	-0.94
5	0.065	0.214	-0.289	0.378	0.234	-0.92
6	0.017	0.192	-0.219	0.384	0.276	-1.15
7	0.042	0.300	-0.358	0.466	0.290	-1.14

Table G.7

G.2 IDENTIFIED MODELS

This appendix contains the identified models for the experimental column work described in Chapter 8, using the input function shown in Graph 8.1.

4th order linear models, corresponding to Graph 8.2 :

$$\begin{aligned} x_D(k) = {} & 0.852690 \times x_D(k-1) + 0.059122 \times x_D(k-2) - 0.111815 \times x_D(k-3) \\ & + 0.042852 \times x_D(k-4) + 0.000242 \times F(k-1) + 0.002868 \times F(k-2) \\ & + 0.001286 \times F(k-3) + 0.000187 \times F(k-4) - 0.002411 \times L_R(k-1) \\ & + 0.030702 \times L_R(k-2) + 0.008200 \times L_R(k-3) - 0.006933 \times L_R(k-4) \\ & + 0.009242 \times Q_S(k-1) - 0.036269 \times Q_S(k-2) - 0.013136 \times Q_S(k-3) \\ & + 0.004974 \times Q_S(k-4) + 0.148041 \end{aligned}$$

$$\begin{aligned} x_W(k) = {} & 0.467639 \times x_W(k-1) + 0.281218 \times x_W(k-2) + 0.089101 \times x_W(k-3) \\ & + 0.052274 \times x_W(k-4) + 0.013798 \times F(k-1) + 0.016735 \times F(k-2) \\ & + 0.002893 \times F(k-3) + 0.006582 \times F(k-4) + 0.004228 \times L_R(k-1) \\ & + 0.007882 \times L_R(k-2) + 0.009282 \times L_R(k-3) + 0.000810 \times L_R(k-4) \\ & - 0.006876 \times Q_S(k-1) - 0.074195 \times Q_S(k-2) - 0.016640 \times Q_S(k-3) \\ & - 0.000779 \times Q_S(k-4) + 0.024794 \end{aligned}$$

4th order diagonal bilinear models, corresponding to Graph 8.3 :

$$
\begin{aligned}
x_D(k) = {} & 0.617553 \times x_D(k-1) + 0.157371 \times x_D(k-2) + 0.091977 \times x_D(k-3) \\
& + 0.082536 \times x_D(k-4) - 0.019867 \times F(k-1) + 0.069744 \times F(k-2) \\
& + 0.082551 \times F(k-3) + 0.033397 \times F(k-4) - 0.081683 \times L_R(k-1) \\
& + 0.049562 \times L_R(k-2) + 0.037725 \times L_R(k-3) + 0.007339 \times L_R(k-4) \\
& - 0.048289 \times Q_S(k-1) - 0.025007 \times Q_S(k-2) - 0.045635 \times Q_S(k-3) \\
& - 0.028103 \times Q_S(k-4) + 0.022290 \times x_D(k-1)F(k-1) \\
& - 0.072534 \times x_D(k-2)F(k-2) - 0.087812 \times x_D(k-3)F(k-3) \\
& - 0.035442 \times x_D(k-4)F(k-4) + 0.086618 \times x_D(k-1)L_R(k-1) \\
& - 0.020585 \times x_D(k-2)L_R(k-2) - 0.029479 \times x_D(k-3)L_R(k-3) \\
& - 0.010922 \times x_D(k-4)L_R(k-4) + 0.062473 \times x_D(k-1)Q_S(k-1) \\
& - 0.011988 \times x_D(k-2)Q_S(k-2) + 0.032753 \times x_D(k-3)Q_S(k-3) \\
& + 0.028533 \times x_D(k-4)Q_S(k-4) + 0.050610
\end{aligned}
$$

$$
\begin{aligned}
x_W(k) = {} & 0.419422 \times x_W(k-1) + 0.347247 \times x_W(k-2) + 0.425675 \times x_W(k-3) \\
& - 0.156992 \times x_W(k-4) + 0.006366 \times F(k-1) + 0.016637 \times F(k-2) \\
& + 0.007010 \times F(k-3) + 0.007542 \times F(k-4) + 0.005883 \times L_R(k-1) \\
& - 0.016426 \times L_R(k-2) + 0.038460 \times L_R(k-3) + 0.002045 \times L_R(k-4) \\
& - 0.005656 \times Q_S(k-1) - 0.051212 \times Q_S(k-2) - 0.010792 \times Q_S(k-3) \\
& - 0.019726 \times Q_S(k-4) + 0.052137 \times x_W(k-1)F(k-1) \\
& + 0.004562 \times x_W(k-2)F(k-2) - 0.028344 \times x_W(k-3)F(k-3) \\
& - 0.000013 \times x_W(k-4)F(k-4) - 0.005388 \times x_W(k-1)L_R(k-1) \\
& + 0.277229 \times x_W(k-2)L_R(k-2) - 0.316784 \times x_W(k-3)L_R(k-3) \\
& + 0.008329 \times x_W(k-4)L_R(k-4) - 0.045414 \times x_W(k-1)Q_S(k-1) \\
& - 0.286929 \times x_W(k-2)Q_S(k-2) - 0.082675 \times x_W(k-3)Q_S(k-3) \\
& + 0.224888 \times x_W(k-4)Q_S(k-4) + 0.013201
\end{aligned}
$$

4th order bilinear models, corresponding to Graph 8.4 :

$$
\begin{aligned}
x_D(k) = {} & 0.508390 \times x_D(k-1) + 0.201771 \times x_D(k-2) + 0.122817 \times x_D(k-3) \\
& + 0.107533 \times x_D(k-4) + 0.015826 \times F(k-1) + 0.054923 \times F(k-2) \\
& + 0.081259 \times F(k-3) + 0.031043 \times F(k-4) - 0.039782 \times L_R(k-1) \\
& + 0.049265 \times L_R(k-2) + 0.020499 \times L_R(k-3) - 0.018001 \times L_R(k-4) \\
& + 0.024042 \times Q_S(k-1) - 0.056109 \times Q_S(k-2) - 0.080706 \times Q_S(k-3) \\
& - 0.075721 \times Q_S(k-4) + 0.021191 \times x_D(k-1)F(k-1) \\
& - 0.037644 \times x_D(k-1)F(k-2) - 0.014439 \times x_D(k-1)F(k-3) \\
& + 0.006586 \times x_D(k-1)F(k-4) - 0.029666 \times x_D(k-2)F(k-1) \\
& - 0.054492 \times x_D(k-2)F(k-2) - 0.063983 \times x_D(k-2)F(k-3) \\
& - 0.035586 \times x_D(k-2)F(k-4) - 0.006227 \times x_D(k-3)F(k-1) \\
& + 0.017474 \times x_D(k-3)F(k-2) - 0.020658 \times x_D(k-3)F(k-3) \\
& - 0.005003 \times x_D(k-3)F(k-4) - 0.001796 \times x_D(k-4)F(k-1) \\
& + 0.018377 \times x_D(k-4)F(k-2) + 0.012506 \times x_D(k-4)F(k-3) \\
& + 0.001206 \times x_D(k-4)F(k-4) + 0.041950 \times x_D(k-1)L_R(k-1) \\
& + 0.021088 \times x_D(k-1)L_R(k-2) + 0.048544 \times x_D(k-1)L_R(k-3) \\
& + 0.075489 \times x_D(k-1)L_R(k-4) + 0.006886 \times x_D(k-2)L_R(k-1) \\
& + 0.003323 \times x_D(k-2)L_R(k-2) - 0.006121 \times x_D(k-2)L_R(k-3) \\
& + 0.002463 \times x_D(k-2)L_R(k-4) - 0.008768 \times x_D(k-3)L_R(k-1) \\
& - 0.023194 \times x_D(k-3)L_R(k-2) - 0.029323 \times x_D(k-3)L_R(k-3) \\
& - 0.030958 \times x_D(k-3)L_R(k-4) + 0.000711 \times x_D(k-4)L_R(k-1) \\
& - 0.021389 \times x_D(k-4)L_R(k-2) - 0.026447 \times x_D(k-4)L_R(k-3) \\
& - 0.029379 \times x_D(k-4)L_R(k-4) + 0.057852 \times x_D(k-1)Q_S(k-1) \\
& + 0.059107 \times x_D(k-1)Q_S(k-2) + 0.071457 \times x_D(k-1)Q_S(k-3) \\
& + 0.058245 \times x_D(k-1)Q_S(k-4) - 0.009743 \times x_D(k-2)Q_S(k-1) \\
& + 0.006175 \times x_D(k-2)Q_S(k-2) + 0.002984 \times x_D(k-2)Q_S(k-3) \\
& + 0.004240 \times x_D(k-2)Q_S(k-4) - 0.024571 \times x_D(k-3)Q_S(k-1) \\
& - 0.009949 \times x_D(k-3)Q_S(k-2) + 0.004827 \times x_D(k-3)Q_S(k-3) \\
& + 0.003944 \times x_D(k-3)Q_S(k-4) - 0.040338 \times x_D(k-4)Q_S(k-1) \\
& - 0.034191 \times x_D(k-4)Q_S(k-2) - 0.005412 \times x_D(k-4)Q_S(k-3) \\
& + 0.013576 \times x_D(k-4)Q_S(k-4) + 0.058855
\end{aligned}
$$

$$
\begin{aligned}
x_W(k) = {} & 0.525073 \times x_W(k-1) + 0.380060 \times x_W(k-2) + 0.049333 \times x_W(k-3) \\
& + 0.036195 \times x_W(k-4) + 0.003874 \times F(k-1) + 0.014717 \times F(k-2) \\
& + 0.011556 \times F(k-3) + 0.012960 \times F(k-4) + 0.005074 \times L_R(k-1) \\
& - 0.023340 \times L_R(k-2) + 0.043366 \times L_R(k-3) + 0.000581 \times L_R(k-4) \\
& - 0.018258 \times Q_S(k-1) - 0.054810 \times Q_S(k-2) + 0.000662 \times Q_S(k-3) \\
& - 0.026129 \times Q_S(k-4) + 0.295205 \times x_W(k-1)F(k-1) \\
& + 0.028766 \times x_W(k-1)F(k-2) - 0.494066 \times x_W(k-1)F(k-3) \\
& - 0.292762 \times x_W(k-1)F(k-4) - 0.055630 \times x_W(k-2)F(k-1) \\
& + 0.313466 \times x_W(k-2)F(k-2) + 0.148916 \times x_W(k-2)F(k-3) \\
& - 0.123489 \times x_W(k-2)F(k-4) - 0.147274 \times x_W(k-3)F(k-1) \\
& - 0.231323 \times x_W(k-3)F(k-2) + 0.226368 \times x_W(k-3)F(k-3) \\
& + 0.036977 \times x_W(k-3)F(k-4) - 0.022357 \times x_W(k-4)F(k-1) \\
& - 0.089310 \times x_W(k-4)F(k-2) + 0.055842 \times x_W(k-4)F(k-3) \\
& + 0.324576 \times x_W(k-4)F(k-4) + 0.381052 \times x_W(k-1)L_R(k-1) \\
& + 0.388760 \times x_W(k-1)L_R(k-2) + 0.093631 \times x_W(k-1)L_R(k-3) \\
& - 0.088543 \times x_W(k-1)L_R(k-4) + 0.019416 \times x_W(k-2)L_R(k-1) \\
& + 0.125690 \times x_W(k-2)L_R(k-2) - 0.226284 \times x_W(k-2)L_R(k-3) \\
& - 0.141301 \times x_W(k-2)L_R(k-4) - 0.196126 \times x_W(k-3)L_R(k-1) \\
& - 0.065757 \times x_W(k-3)L_R(k-2) - 0.146606 \times x_W(k-3)L_R(k-3) \\
& - 0.032109 \times x_W(k-3)L_R(k-4) - 0.198390 \times x_W(k-4)L_R(k-1) \\
& - 0.102231 \times x_W(k-4)L_R(k-2) - 0.073685 \times x_W(k-4)L_R(k-3) \\
& + 0.270887 \times x_W(k-4)L_R(k-4) - 0.244260 \times x_W(k-1)Q_S(k-1) \\
& - 0.042057 \times x_W(k-1)Q_S(k-2) + 0.283560 \times x_W(k-1)Q_S(k-3) \\
& + 0.057584 \times x_W(k-1)Q_S(k-4) - 0.150556 \times x_W(k-2)Q_S(k-1) \\
& - 0.251347 \times x_W(k-2)Q_S(k-2) - 0.159976 \times x_W(k-2)Q_S(k-3) \\
& + 0.180460 \times x_W(k-2)Q_S(k-4) + 0.286750 \times x_W(k-3)Q_S(k-1) \\
& + 0.087089 \times x_W(k-3)Q_S(k-2) + 0.025230 \times x_W(k-3)Q_S(k-3) \\
& + 0.223716 \times x_W(k-3)Q_S(k-4) + 0.198436 \times x_W(k-4)Q_S(k-1) \\
& - 0.046427 \times x_W(k-4)Q_S(k-2) - 0.366820 \times x_W(k-4)Q_S(k-3) \\
& - 0.161399 \times x_W(k-4)Q_S(k-4) + 0.018279
\end{aligned}
$$

APPENDIX H

CASE STUDY PROGRAMS AND DATA

H.1 BILINEAR TANK SIMULATION PROGRAM

```
        PROGRAM TANK
C
C Program to simulate  a bilinear mixing tank with
C a constant heat input.
C
        COMMON/TANKDATA/T1,T2,TIME,FLOW,HEAT
        CHARACTER FILENAME*20
C
C Initialise random number generator
C
        ISEED = 100001
        DO 10 I=1,100
10      ADUMMY = RAN(ISEED)
C
C Initialise tank parameters and variables
C
        TIME = 0.0
        T1 = 40.0
        T2 = 40.0
        TNOISE = 40.0
        TOUT = 40.0
        HEAT = 200
C
C Open output data file
C
        OPEN(UNIT=10,FILE='TANK_DATA.DAT',STATUS='NEW')
C
C Begin simulation loop
C
        DO 20 K=1,3
        IF(K.EQ.1) BASE = 10.0
        IF(K.EQ.2) BASE = 12.5
        IF(K.EQ.3) BASE = 15.0
        DO 20 J=1,40
        CALL GAUSS(ISEED,GRAND)
C
        IF(GRAND.LT.0) THEN
           GRAND = -0.5
        ELSE
           GRAND = 0.5
        END IF
C
        FLOW = BASE + GRAND
        WRITE(10,200)TIME,FLOW,TOUT
200     FORMAT(' ',F10.6,',',F10.6,',',F10.6)
        CALL INTEGRATE
        CALL GAUSS(ISEED,GNOISE)
        TOUT = TNOISE
        TNOISE = T2+GNOISE*0.0
```

```
20      CONTINUE
        WRITE(10,200)TIME,FLOW,TOUT
        END


        SUBROUTINE INTEGRATE
C
C Routine to integrate bilinear tank differential equations using
C a modified Euler method. Integration time consists of fifty
C intervals of 0.01 minutes.
C
        COMMON/TANKDATA/T1,T2,TIME,FLOW,HEAT
C
      DO 10 I=1,50
        TSAVE = T1
        DT1 =.025*HEAT-.025*FLOW*(T1-20.0)
        T1 = T1+.01*DT1
        DT2 = .025*HEAT-.025*FLOW*(T1-20.0)
        T1 = TSAVE+.005*(DT1+DT2)
C
        DT1 = TSAVE-T2
        T2NEW = T2+.01*DT1
        DT2 = T1-T2NEW
        T2 = T2+.005*(DT1+DT2)
C
10      CONTINUE
        TIME=TIME+0.5
        RETURN
        END


        SUBROUTINE GAUSS(ISEED,GNOISE)
C
C Routine to generate Gaussian noise with zero mean and unity
C variance.
C
        CA=2.515517
        CB=0.802853
        CC=0.010328
        DA=1.432788
        DB=0.189269
        DC=0.001308
        A=RAN(ISEED)
        IF(A.GE.0.5) GOTO 1
        B=A
        GOTO 2
1       B=1-A
2       T=(ALOG(1/(B*B)))**0.5
        GNOISE=T-(CA+CB*T+CC*T**2)/(1+DA*T+DB*T**2+DC*T**3)
        IF(A.GE.0.5) GOTO 3
        GNOISE=-GNOISE
3       RETURN
        END
```

H.2 U-D FACTORISATION PROGRAM LISTING

```
        PROGRAM UDUREG
C
C Recursive regression using U-D factorisation method
C
C Reference : Bierman, G.J., "Factorisation methods for
C                  discrete sequential estimation."
C                  Academic Press, 1977.
C
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON/ DATA1 / TIME,FLOW(10),TEMP(10),STEAM(10)
        COMMON/ DATA2 / A(30),Z,X(30)
        COMMON/ DATA3 / N,N_TYPE,N_ORDER,N_DEAD,IFEND
        LOGICAL IFEND
        DIMENSION U(30,30),B(30)
        CHARACTER FILENAME*30
C
C Open input data file
C
        TYPE 101
101     FORMAT (/' ENTER INPUT DATA FILE NAME - ',$)
        READ(5,100) FILENAME
100     FORMAT(A)
        OPEN(UNIT=10,FILE=FILENAME,STATUS='UNKNOWN')
C
C Get model type, order and dead time
C
        TYPE 102
102     FORMAT (/' 1.  LINEAR'/' 2.  DIAGONAL BILINEAR'/
        1       ' 3.  BILINEAR'//' ENTER MODEL TYPE - ',$)
        ACCEPT *, N_TYPE
        TYPE 103
103     FORMAT (/' ENTER MODEL ORDER - ',$)
        ACCEPT *, N_ORDER
        TYPE 104
104     FORMAT (/' ENTER DEAD TIME MEASURED IN SAMPLING',
        1       ' INTERVALS - ',$)
        ACCEPT *, N_DEAD
C
C Calculate parameter vector dimension
C
        IF (N_TYPE .EQ. 1) THEN
           N = 2 * N_ORDER + 1
        ELSE IF (N_TYPE .EQ. 2) THEN
           N = 3 * N_ORDER + 1
        ELSE IF (N_TYPE .EQ. 3) THEN
           N = 2 * N_ORDER + N_ORDER * N_ORDER + 1
        END IF
C
C Initialise measurements
C
        IFEND = .FALSE.
        READ(10,*)TIME,FLOW(1),STEAM(1),TEMP(1)
        DO I=2,10
           FLOW(I)  = FLOW(1)
           TEMP(I)  = TEMP(1)
           STEAM(I) = STEAM(1)
        END DO
```

```
C
C Set forgetting factor and initialise error covariance matrix
C
        DO 92 I=1,N
92      U(I,I) = 10000.0
        FORGET = 1.0
C
C Start of recursive loop
C
91      CALL UPDATA                !Update measurement vector
        IF (IFEND) THEN            !End of data ?
           CALL GAIN_VERIFY        !If yes, start verification
           STOP
        END IF
C
C Update parameter vector
C
        DO 2,J=1,N
2       Z=Z-A(J)*X(J)
        RES=Z
        DO 10 J=N,2,-1
        DO 5 K=1,J-1
5       A(J)=A(J)+U(K,J)*A(K)
10      B(J)=U(J,J)*A(J)
        B(1)=U(1,1)*A(1)
C
        ALPHA=FORGET+B(1)*A(1)
        GAMMA=1./ALPHA
        U(1,1)=GAMMA*U(1,1)
        DO 20 J=2,N
        BETA=ALPHA
        ALPHA=ALPHA+B(J)*A(J)
        RLAMBDA=-A(J)*GAMMA
        GAMMA=1./ALPHA
        U(J,J)=BETA*GAMMA*U(J,J)/FORGET
        DO 20 I=1,J-1
        BETA=U(I,J)
        U(I,J)=BETA+B(I)*RLAMBDA
20      B(I)=B(I)+B(J)*BETA
        Z=Z*GAMMA
        DO 30 J=1,N
30      X(J)=X(J)+B(J)*Z
C
        TYPE 500, TIME,RES,(X(I),I=1,N)
500     FORMAT(' TIME =',F5.1,'RESIDUAL =',F13.8/4(5(' ',F13.8)/))
        GO TO 91
C
C End of recursive loop
C
        END
```

```
      SUBROUTINE UPDATA
C
C Subroutine to update measurement vector
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/ DATA1 / TIME,FLOW(10),TEMP(10),STEAM(10)
      COMMON/ DATA2 / A(30),Z,X(30)
      COMMON/ DATA3 / N,N_TYPE,N_ORDER,N_DEAD,IFEND
      LOGICAL IFEND
C
C Shift measurements back one sampling interval
C
      DO I=0,8
         FLOW(10-I)  = FLOW(9-I)
         TEMP(10-I)  = TEMP(9-I)
         STEAM(10-I) = STEAM(9-I)
      END DOo
C
C Read new measurements
C
      READ(10,*,END=20)TIME,FLOW(1),STEAM(1),TEMP(1)
      Z=TEMP(1)
C
C Evaluate new measurement vector
C
      DO I=1,N_ORDER
         A(I) = TEMP(I+1)
         A(N_ORDER+I) = FLOW(I+1+N_DEAD)
      END DO
C
      IF (N_TYPE .EQ. 2) THEN
         DO I=1,N_ORDER
         A(2*N_ORDER+I) = TEMP(I+1)*FLOW(I+1+N_DEAD)
         END DO
      ELSE IF (N_TYPE .EQ. 3) THEN
         DO I=1,N_ORDER
         DO J=1,N_ORDER
         A(2*N_ORDER+J+(I-1)*N_ORDER)=TEMP(J+1)*FLOW(I+1+N_DEAD)
         END DO
         END DO
      END IF
C
      A(N) = 1
C
      RETURN
20    IFEND = .TRUE.          !Signal end of data
      RETURN
      END
```

```
        SUBROUTINE GAIN_VERIFY
C
C Routine to verify identification results
C
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON/ DATA1 / TIME,FLOW(10),TEMP(10),STEAM(10)
        COMMON/ DATA2 / A(30),Z,X(30)
        COMMON/ DATA3 / N,N_TYPE,N_ORDER,N_DEAD,IFEND
        LOGICAL IFEND
C
C Calculate sums of coefficients of identified model
C
        SUMA = 1.0
        DO I=1,N_ORDER
           SUMA = SUMA - X(I)
           SUMB = SUMB + X(I+N_ORDER)
        END DO
C
        If (N_TYPE .EQ. 2) THEN
           DO I=1,N_ORDER
           SUMC = SUMC + X(2*N_ORDER+I)
           END DO
        ELSE IF (N_TYPE .EQ. 3) THEN
           Do I=1,N_ORDER*N_ORDER
           SUMC = SUMC + X(2*N_ORDER+I)
           END DO
        END IF
C
C Calculate identified model gains for input flowrates of
C 10, 12.5 and 15 l/min
C
        US = 10.0
        YS = (US * SUMB + X(N)) / (SUMA - US * SUMC)
        P_GAIN = (SUMB + YS * SUMC) / (SUMA - US * SUMC)
        TYPE *,'At',US,'&',YS,'gain =',P_GAIN
C
        US = 12.5
        YS = (US * SUMB + X(N)) / (SUMA - US * SUMC)
        P_GAIN = (SUMB + YS * SUMC) / (SUMA - US * SUMC)
        TYPE *,'At',US,'&',YS,'gain =',P_GAIN
C
        US = 15.0
        YS = (US * SUMB + X(N)) / (SUMA - US * SUMC)
        P_GAIN = (SUMB + YS * SUMC) / (SUMA - US * SUMC)
        TYPE *,'At',US,'&',YS,'gain =',P_GAIN
C
C Return to start of input data file, open output data file,
C and initialise measurements
C
        REWIND(10)
        OPEN(UNIT=11,FILE='GNEW.DAT',STATUS='NEW')
        KOUNT=0
        SUMS=0
        READ(10,*,END=150)TIME,FLOW(1),STEAM(1),EXP_TEMP
        US = FLOW(1)
        YS = (US * SUMB + X(N)) / (SUMA - US * SUMC)
        PRE_TEMP = YS
        TEMP(1) = PRE_TEMP
        DO I=2,10
           FLOW(I)  = FLOW(1)
```

```
            TEMP(I)  = TEMP(1)
            STEAM(I) = STEAM(1)
         END DO
C
C Start of recursive loop
C
10       DO I=0,8
            FLOW(10-I)  = FLOW(9-I)
            TEMP(10-I)  = TEMP(9-I)
            STEAM(10-I) = STEAM(9-I)
         END DO
C
         READ(10,*,END=150)TIME,FLOW(1),STEAM(1),EXP_TEMP
         TEMP(1) = PRE_TEMP
C
         PRE_TEMP = 0.0
         DO I=1,N_ORDER
            PRE_TEMP = PRE_TEMP + X(I) * TEMP(I)
            PRE_TEMP = PRE_TEMP + X(I+N_ORDER) * FLOW(I+N_DEAD)
         END DO
C
         IF (N_TYPE .eq. 2) THEN
            DO I=1,N_ORDER
            PRE_TEMP=PRE_TEMP+X(2*N_ORDER+I)*TEMP(I)*FLOW(I+N_DEAD)
            END DO
         ELSE IF (N_TYPE .EQ. 3) THEN
            DO I=1,N_ORDER
            DO J=1,N_ORDER
            PRE_TEMP = PRE_TEMP + X(2*N_ORDER+J+(I-1)*N_ORDER)
         1                                  * TEMP(J)*FLOW(I+N_DEAD)
            END DO
            END DO
         END IF
C
         PRE_TEMP = PRE_TEMP + X(N)
C
         KOUNT = KOUNT+1
         SUMS = SUMS + (TEMP(1) - EXP_TEMP)**2
         WRITE(11,200)TIME,TEMP(1),EXP_TEMP
200      FORMAT(F10.4,',',F10.4,',',F10.4)
         GO TO 10
C
C End of recursive loop
C
C Calculate variance of error between system output and
C identified model output
C
150      VAR=SUMS/(KOUNT+1)
         TYPE *,'VARIANCE = ',VAR
         RETURN
         END
```

H.3 TANK INPUT/OUTPUT DATA

This appendix contains the input/output data of the simulated heat tank system, and corresponds to the input function shown in Graph 6.3.

Time [min]	Inlet water flowrate [l/min]	Outlet water Temperature [°C]
<0.000000	10.000000	40.000000
0.000000	10.500000	40.000000
0.500000	10.500000	40.000000
1.000000	9.500000	39.974533
1.500000	10.500000	39.916130
2.000000	10.500000	39.893528
2.500000	9.500000	39.881557
3.000000	10.500000	39.834827
3.500000	9.500000	39.822231
4.000000	10.500000	39.818935
4.500000	10.500000	39.830326
5.000000	10.500000	39.839485
5.500000	10.500000	39.806046
6.000000	10.500000	39.751553
6.500000	10.500000	39.688507
7.000000	9.500000	39.623947
7.500000	9.500000	39.561718
8.000000	10.500000	39.553391
8.500000	9.500000	39.614975
9.000000	10.500000	39.660980
9.500000	10.500000	39.706039
10.000000	10.500000	39.738995
10.500000	10.500000	39.723064
11.000000	10.500000	39.681896
11.500000	10.500000	39.629299
12.000000	9.500000	39.573174
12.500000	9.500000	39.517891
13.000000	10.500000	39.515305
13.500000	10.500000	39.581650
14.000000	9.500000	39.631741
14.500000	9.500000	39.630119
15.000000	9.500000	39.651108
15.500000	10.500000	39.724148
16.000000	10.500000	39.822029
16.500000	10.500000	39.877853
17.000000	10.500000	39.867367
17.500000	10.500000	39.822117
18.000000	9.500000	39.760571
18.500000	10.500000	39.693317
19.000000	10.500000	39.676239
19.500000	9.500000	39.677338
20.000000	12.000000	39.647442
20.500000	12.000000	39.652493
21.000000	12.000000	39.591728

21.500000	13.000000	39.399689
22.000000	12.000000	39.149647
22.500000	12.000000	38.836380
23.000000	12.000000	38.516670
23.500000	12.000000	38.248379
24.000000	12.000000	38.021626
24.500000	12.000000	37.828983
25.000000	12.000000	37.664719
25.500000	13.000000	37.524261
26.000000	13.000000	37.403938
26.500000	13.000000	37.257484
27.000000	13.000000	37.071457
27.500000	13.000000	36.876011
28.000000	12.000000	36.687233
28.500000	13.000000	36.513027
29.000000	13.000000	36.397457
29.500000	13.000000	36.310890
30.000000	12.000000	36.211601
30.500000	13.000000	36.111607
31.000000	12.000000	36.057343
31.500000	13.000000	36.021969
32.000000	13.000000	36.006332
32.500000	12.000000	35.993916
33.000000	13.000000	35.951550
33.500000	12.000000	35.936287
34.000000	13.000000	35.928047
34.500000	13.000000	35.931824
35.000000	12.000000	35.933704
35.500000	13.000000	35.902267
36.000000	12.000000	35.895473
36.500000	13.000000	35.893959
37.000000	12.000000	35.903156
37.500000	13.000000	35.909496
38.000000	13.000000	35.921886
38.500000	12.000000	35.928818
39.000000	13.000000	35.900269
39.500000	12.000000	35.895088
40.000000	15.500000	35.894421
40.500000	15.500000	35.904034
41.000000	15.500000	35.811207
41.500000	15.500000	35.563286
42.000000	14.500000	35.255054
42.500000	14.500000	34.938042
43.000000	15.500000	34.674217
43.500000	15.500000	34.483253
44.000000	15.500000	34.306366
44.500000	14.500000	34.120735
45.000000	14.500000	33.943604
45.500000	14.500000	33.816933
46.000000	14.500000	33.751289
46.500000	15.500000	33.720795
47.000000	14.500000	33.710079
47.500000	15.500000	33.675900
48.000000	14.500000	33.639122
48.500000	15.500000	33.596626
49.000000	14.500000	33.560978
49.500000	14.500000	33.524326
50.000000	15.500000	33.496460
50.500000	15.500000	33.501698
51.000000	15.500000	33.489578

51.500000	15.500000	33.442890
52.000000	15.500000	33.382149
52.500000	15.500000	33.318611
53.000000	14.500000	33.258076
53.500000	14.500000	33.203232
54.000000	14.500000	33.187790
54.500000	15.500000	33.219471
55.000000	14.500000	33.272915
55.500000	15.500000	33.300438
56.000000	15.500000	33.319912
56.500000	15.500000	33.327190
57.000000	15.500000	33.301437
57.500000	14.500000	33.260975
58.000000	14.500000	33.215961
58.500000	15.500000	33.204624
59.000000	14.500000	33.237278
59.500000	14.500000	33.256752
60.000000	14.500000	33.275116

H.4 IDENTIFIED MODELS

This appendix contains the identified models for the heated tank case study described in Chapter 6, using the input function show in Graph 6.3.

1st order linear model, corresponding to Graph 6.4 :

$$T_o(k) = 0.939283 \times T_o(k-1) - 0.082821 \times F(k-2) + 3.219775$$

2nd order linear model, corresponding to Graph 6.5 :

$$T_o(k) = 1.591789 \times T_o(k-1) - 0.640826 \times T_o(k-2) - 0.040247 \times F(k-2) - 0.024143 \times F(k-3) + 2.593271$$

3rd order linear model, corresponding to Graph 6.6 :

$$T_o(k) = 2.384869 \times T_o(k-1) - 1.857147 \times T_o(k-2) + 0.468645 \times T_o(k-3) - 0.042264 \times F(k-2) + 0.005290 \times F(k-3) + 0.032472 \times F(k-4) + 0.188765$$

1st order bilinear model, corresponding to Graph 6.7 :

$$T_o(k) = 1.050094 \times T_o(k-1) + 0.237320 \times F(k-2) - 0.008651 \times T_o(k-1)F(k-2) - 0.922648$$

2nd order diagonal bilinear model, corresponding to Graph 6.8 :

$$T_o(k) = 1.487331 \times T_o(k-1) - 0.485058 \times T_o(k-2) + 0.050022 \times F(k-2) + 0.047151 \times F(k-3) - 0.002493 \times T_o(k-1)F(k-2) - 0.002196 \times T_o(k-2)F(k-3) + 0.813350$$

3rd order diagonal bilinear model, corresponding to Graph 6.9 :

$$T_o(k) = 1.689288 \times T_o(k-1) - 0.782535 \times T_o(k-2) + 0.094911 \times T_o(k-3) + 0.052511 \times F(k-2) + 0.039520 \times F(k-3) - 0.014953 \times F(k-4) - 0.002562 \times T_o(k-1)F(k-2) - 0.001760 \times T_o(k-2)F(k-3) + 0.000592 \times T_o(k-3)F(k-4) + 0.654666$$

2nd order bilinear model, corresponding to Graph 6.10 :

$$T_o(k) = 1.614812 \times T_o(k-1) - 0.614009 \times T_o(k-2) + 0.051743 \times F(k-2) + 0.041390 \times F(k-3) - 0.007011 \times T_o(k-1)F(k-2) + 0.004458 \times T_o(k-2)F(k-2) - 0.005279 \times T_o(k-1)F(k-3) + 0.003230 \times T_o(k-2)F(k-3) + 0.877258$$

3rd order bilinear model, corresponding to Graph 6.11 :

$$T_o(k) = 1.267166 \times T_o(k-1) - 0.054203 \times T_o(k-2) - 0.212222 \times T_o(k-3) + 0.055131 \times F(k-2) + 0.062737 \times F(k-3) + 0.011021 \times F(k-4) - 0.009230 \times T_o(k-1)F(k-2) + 0.007961 \times T_o(k-2)F(k-2) - 0.001378 \times T_o(k-3)F(k-2) - 0.011736 \times T_o(k-1)F(k-3) + 0.011959 \times T_o(k-2)F(k-3) - 0.003307 \times T_o(k-3)F(k-3) + 0.004941 \times T_o(k-1)F(k-4) - 0.010868 \times T_o(k-2)F(k-4) + 0.005264 \times T_o(k-3)F(k-4) + 1.239539$$

2nd order diagonal bilinear model with overestimated dead time, corresponding to Graph 6.12 :

$$T_o(k) = 1.540087 \times T_o(k-1) - 0.526065 \times T_o(k-2) + 0.116211 \times F(k-3) - 0.002044 \times F(k-4) - 0.004474 \times T_o(k-1)F(k-3) - 0.000257 \times T_o(k-2)F(k-4) + 0.185215$$

3rd order diagonal bilinear model with overestimated dead time, corresponding to Graph 6.13 :

$$T_o(k) = 1.066292 \times T_o(k-1) + 0.105406 \times T_o(k-2) - 0.153443 \times T_o(k-3) + 0.111702 \times F(k-3) - 0.020372 \times F(k-4) + 0.089174 \times F(k-5) - 0.004893 \times T_o(k-1)F(k-3) - 0.000279 \times T_o(k-2)F(k-4) - 0.002585 \times T_o(k-3)F(k-5) + 0.563532$$

2nd order diagonal bilinear model with underestimated dead time, corresponding to Graph 6.14 :

$$T_o(k) = 1.649590 \times T_o(k-1) - 0.652735 \times T_o(k-2) - 0.025028 \times F(k-1) + 0.076119 \times F(k-2) + 0.000540 \times T_o(k-1)F(k-1) - 0.003263 \times T_o(k-2)F(k-2) + 0.703104$$

3rd order diagonal bilinear model with underestimated dead time, corresponding to Graph 6.15 :

$$T_o(k) = 1.461537 \times T_o(k-1) - 0.492666 \times T_o(k-2) + 0.033791 \times T_o(k-3) - 0.002089 \times F(k-1) + 0.050970 \times F(k-2) + 0.048809 \times F(k-3) + 0.000062 \times T_o(k-1)F(k-1) - 0.002516 \times T_o(k-2)F(k-2) - 0.002229 \times T_o(k-3)F(k-3) + 0.789244$$

2nd order diagonal bilinear model with measurement noise (SD = 0.01°C), corresponding to Graph 6.16 :

$$T_o(k) = 1.403482 \times T_o(k-1) - 0.398857 \times T_o(k-2) + 0.052088 \times F(k-2) + 0.062239 \times F(k-3) - 0.002614 \times T_o(k-1)F(k-2) - 0.002788 \times T_o(k-2)F(k-3) + 0.832443$$

2nd order diagonal bilinear model with measurement noise (SD = 0.1°C), corresponding to Graph 6.17 :

$$T_o(k) = 0.630123 \times T_o(k-1) + 0.411904 \times T_o(k-2) + 0.238017 \times F(k-2) + 0.086560 \times F(k-3) - 0.007869 \times T_o(k-1)F(k-2) - 0.005615 \times T_o(k-2)F(k-3) + 0.467235$$

2nd order diagonal bilinear model with measurement noise (SD = 0.1°C) and three times as many data points, corresponding to Graph 6.18 :

$$T_o(k) = 0.711177 \times T_o(k-1) + 0.289086 \times T_o(k-2) + 0.183440 \times F(k-2) + 0.027385 \times F(k-3) - 0.006410 \times T_o(k-1)F(k-2) - 0.004313 \times T_o(k-2)F(k-3) + 2.182950$$

2nd order diagonal bilinear model with measurement noise (SD = 0.1°C) and ten times as many data points, corresponding to Graph 6.19 :

$$T_o(k) = 0.774633 \times T_o(k-1) + 0.235461 \times T_o(k-2) + 0.207447 \times F(k-2) + 0.020038 \times F(k-3) - 0.007335 \times T_o(k-1)F(k-2) - 0.003651 \times T_o(k-2)F(k-3) + 1.738320$$